Technical University of Crete, Greece
Department of Electronic and Computer Engineering

Lineage Processing in Uncertain Operator Pipelines

Lampros C. Papageorgiou

Thesis Advisor
Professor Minos Garofalakis

Thesis Committee
Professor Minos Garofalakis
Assistant Professor Michail Lagoudakis
Assistant Professor Antonios Deligiannakis

Chania, July 2011

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμημα Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

Επεξεργασία Προέλευσης Δεδομένων σε Δίκτυα Αβέβαιων Τελεστών

Λάμπρος Κ. Παπαγεωργίου

Επιβλέπων Καθηγητής:
Καθηγητής Μίνως Γαροφαλάκης

Εξεταστική Επιτροπή:
Καθηγητής Μίνως Γαροφαλάκης
Επίκ. Καθηγητής Μιχαήλ Λαγουδάκης
Επίκ. Καθηγητής Αντώνιος Δεληγιαννάκης

Χανιά, Ιούλιος 2011

Acknowledgement

Several people played an important role in the accomplishing of this diploma thesis.

First and foremost, I am heartily thankful to my supervisor, Professor Minos Garofalakis, whose encouragement, guidance and support from the initial to the final level enabled me to develop a deep understanding of the subject. I feel much indebted to him for both inspiring me and for giving me the opportunity to work on this very interesting field of research.

Also, I would like to thank Assistant Professors Michail Lagoudakis and Antonios Deligiannakis for agreeing to evaluate my diploma thesis. Moreover, I would like to thank them for all the precious knowledge I have gained by them.

I offer my regards and blessings to all of my friends, and especially Evaggelos Vazaios, who supported me in any respect during the completion of this thesis.

Most of all, I would like to thank my family for their enormous help, understanding and support throughout all these years as a student. Following their paradigms of excellence has inspired me to achieve my own goals.

Abstract

The use of pipelined operators to manage data coming from web extraction tasks is a typical strategy in most Community Information Management (CIM) platforms. Commonly, web extraction results are inherently uncertain. Incorporating operators coming from the Machine Learning community in such pipelines would help the platform adapt to various domains and improve over time; however, it also adds another source of uncertainty: the intermediate uncertain operators themselves. In this work, we propose a method which uses the lineage of the returned results in order to quantify the influence of not only the input data, but also the influence of the uncertain intermediate operators to the returned results. Moreover, we have developed exact as well as approximate techniques to efficiently repair pipelines populated with uncertain operators by returning a small fraction of the operators which, when refined, would improve the pipeline results. Our definitions and methodology generalize the influence definition of Re and Suciu, extending it to apply to pipelines populated with uncertain operators. Furthermore, we have implemented our approach as an extension to PostgreSQL and tested it on various pipelines. Our experimental results have shown that our approach successfully identifies the top-k influential operators of an extraction pipeline, and provides high quality results compared to other approaches, while maintaining low cost.

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview and Problem Statement

The role of modern web communities to inform their members about everything new that is written or said that may be of their interest imposes the need to build efficient and scalable community information management (CIM) platforms. CIM platforms (e.g. Purple Sox [4], Cimple [14]) commonly use extraction and integration pipelines to gather data from web sources, semantically categorize this data, and return suggestions to their users. A variety of systems is used in order to extract data from the web and provide possibly interesting suggestions of web pages for the community members. Among these suggestions, the quality and accuracy of the returned results can range from totally irrelevant web pages to others of utmost interest.

Given that the data feeding such pipelines comes from a web extraction process, this data is inherently uncertain. A common approach is to use a probabilistic database to store the extracted data that feeds the pipeline. In addition, the need to integrate this data efficiently would be satisfied by incorporating operators from the Machine Learning community in a pipeline. Such operators learn from and adapt to a wide variety of domains and data sources. Our work introduces a novel model that incorporates uncertain operators in extraction pipelines. Although such operators are appropriate for the needs of CIM platforms, their output is also uncertain. As a result, in addition to the uncertainty introduced by the extracted input data itself, we identify another source of uncertainty in

such pipelines, i.e., the uncertainty introduced by the operators used to integrate this data. Considering uncertain operators, results in a significant increase in the complexity of the lineage processing and the debugging of such a system.

One of the main goals of this work is to develop a method that would make such data *explainable*, in the sense that the history of extraction is tracked by the system. Returning information about the origin of data (lineage) is a critical task, as it will help users understand which stage of integration caused ambiguous or faulty results. It is then clear that lineage processing and efficient debugging and improvement of such pipelines is a great challenge.

Empowering engaged users such as community moderators or even common users to participate in and repair the information integration process, will improve future suggestions. Over time, feedback from users can be used to improve the extraction and integration process, requiring less expert activity and knowledge and allowing the quality of the extracted information to be improved until only the latest and greatest information about the topic of their interest is returned.

For example, if the information extracted about a particular topic is incorrect, the community members could simply click and flag that data -ultimately allowing the system to assign greater confidence to higher quality extraction results and improve the method used, which resulted in faulty suggestions.

Suppose that we are provided with a probabilistic database with uncertain data and an extraction pipeline populated with uncertain operators. Our work focuses on locating those operators which have the greatest impact to the result. Finding a small set of *influential* nodes is imperative to perform a low-cost repair of the pipeline.

1.2 Motivating example

Our motivating example is the scenario of a CIM platform which manages a large number of sophisticated extraction and integration hierarchical pipelines across different application domains, at web scale and with minimum human involvement. Such a system should offer full functionality on three key aspects:

extensibility, explainability and social feedback support. Especially for the explainability and social feedback support -the focus of our work- the system should offer the ability to track the lineage of extraction results.

Moreover, such a system system should provide both the users and developers with possible sources of error, facilitating the process of efficiently repairing the pipeline. It should also be able to accept and effectively manage intermittent and noisy social feedback on the quality of the extracted data. As a result, such a framework will transform users from sheer observers to critical components of the system and active critics of the data that will be suggested to all community members in the future.

The most interesting examples of CIM platforms are the following:

- **PurpleSOX:** Purple SOX (Socially Oriented eXtraction) [4] is a prototype extraction management system (EMS) by Yahoo! Research. Purple SOX focuses on technologies to extract and manage structured information from the Web related to a specific community.

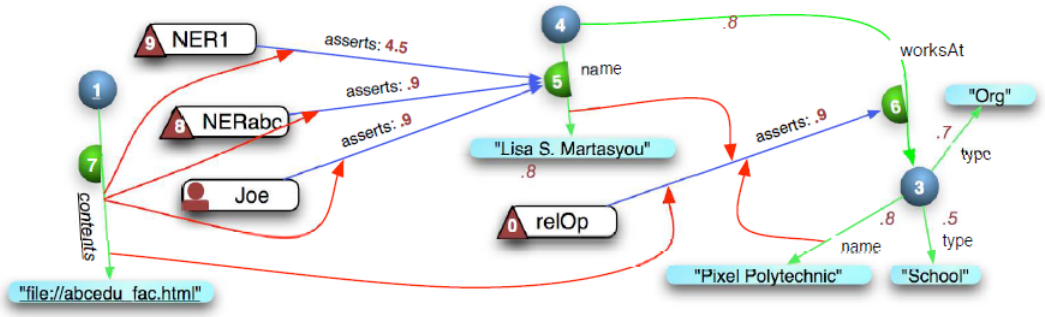


Figure 1.1: Provenance and chained inference pipeline example of PSOX

Figure 1.1 shows an example of a Purple SOX pipeline. The blue nodes (1,3,4) denote extracted entities and the green (5,6,7) and crimson nodes (0,8,9) denote the uncertain operators. The red edges denote the operators inputs, which may be some aforementioned entities or the output of other operators in order to, e.g., classify entities into categories. In Section 3.3.2 we provide further details about the uncertain operators and the pipelines which are described by our model.

- **Cimple:** Cimple is a joint project between the University of Illinois, University of Wisconsin-Madison and Yahoo! Research. It develops a software platform that can be rapidly deployed and customized to manage data-rich online communities [14]. To drive and validate Cimple, DBLife was built, a prototype system that manages information for the database research community [15]. DBLife aggregates structured information about the database community from data on the Web.

1.3 Contributions and Prior Work

Our goal is to efficiently capture the lineage of the output results of an extraction pipeline populated with uncertain operators, quantify the influence of each uncertain operator, and provide a mechanism to repair the pipeline and improve the quality of the returned results, while maintaining the lowest possible cost.

An active area of research which addresses the problem of quantifying the influence among uncertain data is the work on lineage processing in probabilistic databases. However, most of the prior art in this area, i.e. [1],[21],[23],[24], only deals with *data uncertainty*. The term data uncertainty refers to the fact that the whole system's uncertainty derives from the uncertainty of the input data. For example, Re and Suciu [1] define the influence of independent uncertain input data to a query result in a probabilistic database setting. Although this approach seems to be very useful for the problem we have addressed, their definitions are not applicable to our setting since their approach considers arbitrary deterministic operators to manage this uncertain data. Due to the fact that the uncertainty of the whole system in [1] derives from the input data only, adopting their approach to our problem would ignore the uncertainty introduced by the operators, leading to erroneous conclusions. In fact, our definitions and methodology generalize the definition of influence in [1], making it possible to identify the influence in pipelines with uncertain operators.

Other approaches that deal with the improvement of information extraction pipelines do not consider uncertainty at either the data or the operators at all. Such a piece of work is PROBER (Provenance-Based Debugger) [3]. PROBER is a generic framework for debugging information extraction pipelines composed

of arbitrary ("black-box") operators. Although the problem we handle has similarities to PROBER (the common goal is to find sources of errors in pipelines of operators that manage extracted data), the qualitative characteristics of the two settings are very different. PROBER refers to a deterministic setting, so there is no uncertainty at either the input data or the intermediate operators of the pipeline. As a result, PROBER examines the lineage of (incorrect) output records only referring to input tuples that impacted this output record. We extend their approach by not only considering uncertain inputs and intermediate operators, but we also quantify each operator's influence to the root's output. A more thorough description of PROBER is given in the next chapter.

To the best of our knowledge, there is no work addressing the case in which uncertain operators are incorporated into information extraction and integration pipelines. For this reason, we introduce the term *operator uncertainty* in such pipelines. As expected, there is no prior work on quantifying the influence of each uncertain operator in such a setting.

An additional contribution of this thesis is that our definitions and methodology generalize the influence definition provided by Re and Suciu [1]. We extend their definition to pipelines with uncertain intermediate operators, consider their impact to the lineage, and quantify their influence to the returned results.

Moreover, our approach provides an operator refinement plan in order to gain maximum benefit in terms of the quality of the returned results, while refining only a small fraction of all pipeline operators. We argue that we can achieve comparably high improvement of the quality of results as if we refined all pipeline's operators.

Ikeda and Widom [7] provide a thorough description of open problems in the area of data lineage. Our work efficiently handles and solves a number of such problems: We handle the problem of "how-lineage", that is, *given some output, how were the inputs manipulated to produce the output?* Moreover, we efficiently trace possible sources of error. We take into account and examine both incorrect input data and incorrect transformations, i.e. erroneous intermediate values that come from uncertain operators. Furthermore, we apply corrections that will efficiently propagate forward and bring the whole system in a more correct and certain state.

To sum up, the main contributions of this work are as follows:

- We investigate a novel lineage problem setting that incorporates operators from the machine learning field into a pipeline that performs information extraction tasks, and feeds these uncertain operators with input coming from probabilistic databases.
- We represent the lineage function of a returned result by also taking into account the effects of the uncertain operators and not only the input data.
- We present a method for quantifying the *influence* of each operator to the returned results and a metric for the *blame* we put on each influential operator for an erroneous result. Our definitions and methodology generalize the definition of influence provided by Ré and Suciu [1], extending them to handle uncertain operators.
- We present an efficient method for low-cost improvement of such pipelines, using the definitions of influence and blame. This provides a benefit which outperforms the naive approach of refining all pipeline operators. We shrewdly choose to refine only a small fraction of critical operators.
- We provide high quality experimental results which show the efficiency of our approach while maintaining low execution time.

The remainder of this thesis is organized as follows. In Chapter 2, we give a short description of the work in various related fields. Chapter 3 provides the setting of our problem and some basic notions. In Chapter 4 we describe and explain our approach. Finally, in Chapter 5 we provide and discuss our implementation and the experimental results.

Chapter 2

Background and Related Work

Our work relates to and borrows from the state-of-the-art in numerous fields, such as: probabilistic data management, lineage processing, causality. Here we provide some short description of each field, some basic definitions, and how each field relates to our work.

2.1 Probabilistic Databases

A wide range of applications have recently emerged the need to manage large, imprecise data sets. The reasons for imprecision in data are as diverse as the applications themselves: in sensor and RFID data, imprecision is due to measurement errors; in information extraction, imprecision comes from the inherent ambiguity in natural-language text; and in business intelligence, imprecision is used to reduce the cost of data cleaning. Imprecise data has no place in traditional, precise database applications, and so, current database management systems are not prepared to deal with it. In contrast, these newly emerging applications offer value precisely because they query, search, and aggregate large volumes of imprecise data.

In information extraction, we usually face the problem of outdated/erroneous information which is often present on the Web. Even if an extractor is operating on an up-to-date web page, the difficulty of the extraction problem forces the extractors to produce many alternative extractions or risk missing valuable data.

Thus, each attribute may contain several possible values; or equivalently, one can think of each row of a table as being a separate uncertain tuple.

Sen et al. [25] consider the problem of uncertain data representation, and Dalvi et al. [17] survey the state of the art techniques to handle imprecise data which models imprecision as probabilistic data and provide a concise description of the use of probabilistic databases in uncertain data management. As they point out, there are two constraints on this data: tuples with different values of a key attribute are independent; and tuples with the same key attribute but difference in another attribute are mutually exclusive. Should we assign probabilities to such data, the former refers to *tuple-existence uncertainty*, where we are not able to ascertain if a tuple should be included in the database or not, and the latter refers to *attribute-value uncertainty*, where we are unable to ascertain if an attribute value -or which attribute value- is correct and thus we assign a probability of it being correct. In both cases, the uncertainty in the data is represented as a probabilistic confidence score. The following definition of a probabilistic database refers to tuple-existence uncertainty. Attribute-value uncertainty is handled similarly.

According to the definition of Suciu [16], a probabilistic database is a database in which every tuple t belongs to the database with some probability P(t); when P(t) = 1 then the tuple is certain to belong to the database; when 0<P(t)<1 then it belongs to the database only with some probability; when P(t) = 0 then the tuple is certain not to belong to the database, and it is not necessary to bother representing it. A traditional (deterministic) database corresponds to the case when P(t) = 1 for all tuples t.

Figure 2.1 shows the difference between the two above approaches. Table HasObject describes the attribute-value uncertainty, and table Meets describes the tuple-existence uncertainty.

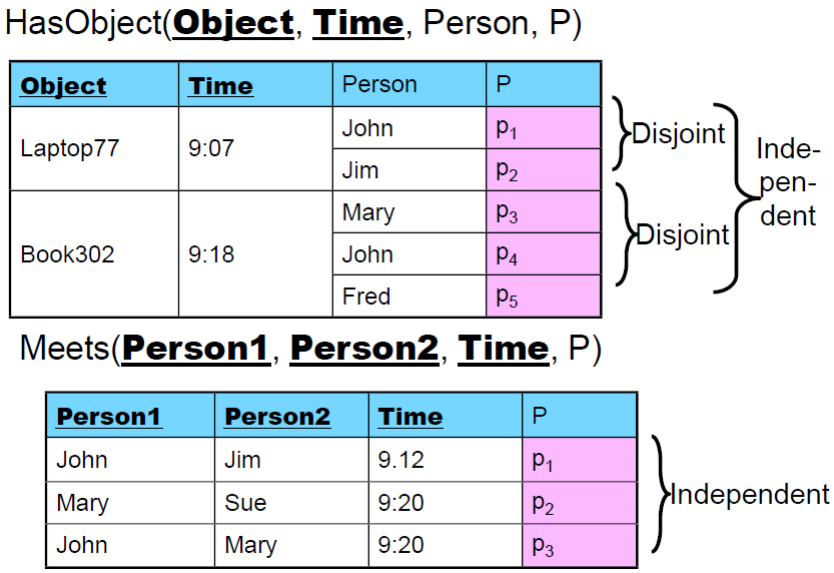


Figure 2.1: Attribute-value and tuple-existence uncertainty

Probabilistic databases are designed to allow uncertain data to be managed directly by a relational database system. Several types of applications have been considered. Specifically in Information Extraction the goal is to extract structured data from a collection of unstructured text documents. Examples include address segmentation, citation extraction, extractions for comparison shopping, hotels, restaurant guides, etc. The schema is given in advance by the user, and the extractor is tailored to that specific schema. All approaches to extraction are imprecise, and most often can associate a probability score to each item extracted. A probabilistic database allows these probability scores to be stored natively.

In this thesis, we consider a probabilistic database as an input data source: we assume that web extracted data are stored in a probabilistic database and the pipelines we describe are fed with this input. The uncertainty in the data is represented as a probabilistic confidence score, which is computed by the extractor. Uncertain data is annotated with a confidence score, which is interpreted as a probability. For example, Conditional Random Fields produce extractions with semantically meaningful confidence scores. Other sources of uncertainty can also be converted to confidence scores, for example probabilities produced by entity matching algorithms or Support Vector Machines.

2.2 Possible Worlds Semantics

The possible world semantics in probabilistic databases, introduced by Dalvi and Suciu [20] and also described in [1],[17],[19],[22] provide all possible states of a probabilistic database.

Possible worlds are a means of expressing uncertainty. Under possible worlds semantics, a probabilistic database is viewed as encoding a probability distribution over all possible deterministic instances of the database.

A natural way to deal with uncertain data is to interpret all uncertainties in the system as probabilistic values in the interval [0,1], and then apply a structured query paradigm that ranks results by their relevance as in common probabilistic databases (PDBs).

Following the definition of Dalvi et al. [17] in order to describe probabilistic databases using possible worlds semantics, it is:

A probabilistic database is a discrete probability space PDB=(W,$\mathbf{P}$), where W=$\{I_1,I_2,...,I_n\}$ is a set of possible instances, called possible worlds, and $\mathbf{P}$:W$\rightarrow$[0,1] is such that $\sum_{j=1,n} \mathbf{P}(I_j)=1$.

The probability that some tuple t belongs to a randomly chosen world is $\mathbf{P}(t)=\sum_{j:t\in I_j} \mathbf{P}(I_j)$, and is also called the marginal probability of the tuple t.

Conceptually, one of the possible worlds is "true", but we do not know which one (subjectivist Bayesian interpretation).

- Each possible world is identified by a valuation that assigns one of the possible values to each variable.

- The probability of the possible world is the product of weights of the values of the variables, assuming independence.

We have chosen to ground our framework to possible worlds semantics, as it is a widely accepted, simple and concise method, used in probabilistic databases and query evaluation.

2.3 Data Lineage

According to the definition of Gupta [26], the term "data lineage" refers to a record trail that accounts for the origin of a piece of data (in a database, document or repository) together with an explanation of how and why it got to the present place.

In computer science, lineage -also called provenance- describes the source and derivation of data. Lineage, or provenance, in its most general form describes where data came from, how it was derived, and how it was updated over time. A knowledge of where a data element has come from is essential in assessing the quality of the database and thus, a record of provenance is essential to the trust one places in data.

The lineage of data has recently been recognized as central to the trust one places in data. It is also important to annotation, to data integration and to probabilistic databases.

In addition, lineage has recently been shown to be important to understanding the transport of annotation in database views, to data integration, to view update and maintenance, and to probabilistic databases. Information management systems today exploit lineage in tasks ranging from data verification in curated databases to confidence computation in probabilistic databases.

Although lineage can be very valuable for applications, storing and querying lineage can be expensive.

Following Dalvi et al. [17], lineage also provides a powerful mechanism for understanding and resolving uncertainty. With lineage, user feedback on correctness of results can be traced back to the sources of relevant data, allowing unreliable sources to be identified. Users can provide much detailed feedback if data lineage is made visible to them. For example, in information extraction applications where data items are generated by pipelines of AI operators, users can not only indicate if a data item is correct, but can look at the lineage of data items to locate the exact operator making the error. The above need in lineage processing, as described in [17] is exactly what our work intends to accomplish.

2.4 Causality

Lineage of query results is closely related to causality, in cases of why-lineage and where-lineage([8],[9]). As discussed by Meliou et al. [8], causality in databases aims to answer the following question: given a query over a database instance and a particular output of the query, which tuple(s) in the instance caused that output to the query?

When analyzing data sets, users are often interested in the causes of their observations: "What caused my personalized newsfeed to contain more than 10 items related to volcanos?". Database research that addresses these or similar questions is mainly work on lineage of query results, such as why or where lineage, and very recently, explanations for non-answers. While these approaches differ over what the response to such questions should be, all of them seem to be linked through a common underlying theme: understanding causal relationships in databases.

Causal relationships cannot be explicitly modelled in current database systems, which offer no specific support for such queries. Mining techniques can infer statistically significant data patterns but they are not sufficient to draw conclusions, as *correlation does not necessarily imply causation*. A future goal would be to extend the capabilities of current database systems by incorporating causal reasoning. This will allow databases to model causal dependencies, and users to issue queries that can interpret them to provide explanations for their observations. Starting from the very basic functionality of justifying the presence or absence of results for a given query, causality-enabled databases can find many practical applications.

Our work aims at modelling the causal relationships of not only the input data of an uncertain operator pipeline, but also the causal relationship of each intermediate operator to the returned result. This allows us to provide all possible explanations for every observed output, extending the domain of possible causes by also taking into account the intermediate operators. As a result, we transform the role of the intermediate uncertain operators from sheer information transferring components to critical components of the system, since an uncertain operator may be the cause of an observation.

2.5 Influence

Various definitions of influence -sometimes also called responsibility- can be found in many publications, both in the field of probabilistic databases and in the field of causality. Much of this literature has been studied thorough, throughout the design and implementation of this thesis. Here, we provide a short summary for each of the most interesting publications:

- Re and Suciu [1] provide a definition of influence in a probabilistic relational database setting. In a probabilistic database, base tuple probabilities are independent; these base tuples are represented by atoms. The main idea is that some atoms are more influential than others. Informally, an atom is influential if there are many assignments of all other atoms such that the atom under consideration is the deciding vote, i.e., changing the assignment of the influential atom changes whether tuple t is returned.

  The influence of an atom i is defined by the following formula:

  $$\mathrm{Inf}_i(\lambda_t) \stackrel{\mathrm{def}}{=} \mathbf{P}[\lambda_t(\mathrm{A}) \neq \lambda_t(\mathrm{A} \oplus \{\mathrm{i}\})]$$

  where A the set of atoms (inputs) and λ_t the lineage function of the returned tuple t. If λ_t evaluates to True, then t is returned by the query.

  According to their definition, the probability that atom i determines whether a tuple t is present in the output, is the probability that all other atoms A output a value, for which the atom under consideration becomes the deciding vote. In short, they define the influence of an atom i as the total probability of all assignments of the rest atoms for which i determines t's presence at the output.

  In comparison to this work, our approach does not consider only input data uncertainty. Our definitions and methodology generalize their definition of influence by also considering operator uncertainty and thus enabling to quantify the influence introduced by the uncertain operators.

- In PROBER (Provenance-Based Debugger) [3], the definition of the provenance model is based on minimal subsets of operator inputs -MISets- to link each output record with a minimal set of contributing input records. Intuitively, an MISet gives the fewest input records required for a particular output record r to be present. Therefore, an MISet provides users with one possible reason for the occurrence of r. In order to measure the influence of each input, they define the impact of a system input i that results in system output r, as the number of participations of i in all MISets that output r. In simple words, they count how many MISets that result in output r include i.

  The setting of this paper shares similarities with ours. Their main goal is to track the provenance of the output, as the user may be interested in understanding why certain incorrect records were generated and then identify and eliminate their sources. However, there are two big differences: First, in PROBER there is no uncertainty, i.e., both the inputs and the operators are deterministic; and, second, PROBER considers only the input data as possible causes of erroneous results.

- In a recently published paper, Kanagal et al. [21] define the notions of influence and explanations in a probabilistic database context. Their work aims to enhance the information provided by the lineage of the returned tuples, by justifying about how input tuples are responsible for certain outputs. The definition of influence provided is a generalization of the definition of influence by Re et al. [1], since it is applicable for handling a larger variety of queries. The goal of this work is to use the lineage for extracting two useful entities, i.e., the most influential/sensitive input tuples and the best explanations from the lineage formula and the associated input tuple probabilities.

  Overall, this paper provides a thorough study on finding influential tuples and explaining query results in a probabilistic database context. However, only data uncertainty is considered whereas our work also considers operator uncertainty.

- Chockler et al. [2] introduce the definition of responsibility in a causal model. This is one of the first papers in the area of causality -along with Halpern and Pearl [11]- that defines *responsibility*. Responsibility measures the minimal number of changes that have to be made in a set of causes in order to make an effect counterfactually depend on some cause. The term "counterfactual" can be better understood by the following statement: "If A had not happened, then B would not have happened". Then, we say that B counterfactually depends on A.

  The notion of responsibility can be more easily understood via the example of voters: let us consider an election with 11 voters and two outcomes, 11-0 and 6-5. In the first case, each voter has degree of responsibility of $\frac{1}{6}$, since 5 changes have to be made before a vote becomes critical. In the second case, the degree of responsibility of each voter who voted for the winner is 1, since only one voter is needed to change her vote in order to change the outcome; each voter is critical. The degree of responsibility of the voters who voted for the loser is 0, since a change in their vote cannot change the outcome.

  This definition, as well as the closely related definition of Halpern and Pearl [11], can be considered as the basis of the responsibility definition in a database setting for the following two papers.

- Meliou et al. [5] propose causality as a unified framework to explain query answers and non-answers. Extending the work of [11], they introduce functional causes as a refined definition of causality, in order to model provenance as well as define explanations in a database context and give graded degrees of responsibility to individual causes. Responsibility is a measure for degree of causality, first introduced by Chockler and Halpern in [2]. Meliou et al. in this paper provide a definition of responsibility in causal networks, redefined here for functional causes. Practically, in order to compute the degree of responsibility for a tuple t, one must find the smallest set of tuples that, when inverted (i.e., either inserted or deleted), make tuple t counterfactual for the condition.

This paper provides the theoretical foundations of causality theory in the database context. However, the paper focuses on deterministic cases, i.e., neither data nor operator uncertainty are considered.

- In a follow-up paper, Meliou et al. [8] initiate a discussion on causality in databases and its relation to provenance.

  Causality in databases aims to answer the following question: given a query over a database instance and a particular output of the query, which tuple(s) in the instance caused that output to the query? Similar to Halpern and Pearl's [11] definition of causality, the above question also expresses the notion of intervention: what happens to the outcome if we change the state of the input. As a starting point, this work proposes to define causality in databases by using the lineage of the answer to a query. The responsibility of an input tuple t is a function of the minimal number of tuples that we need to remove from the database before t becomes counterfactual.

  This work provides very good "entry-level" information on causality, and its relation to lineage and databases. However, there is no connection to either probabilistic databases or data and operator uncertainty.

Having presented all related influence and responsibility definitions by the related work, we may draw the following conclusions:

i) there is no related work that performs influence quantification under a both data and operator uncertainty setting, and

ii) most definitions are closely related to each other, meaning that there is a general approach for quantifying influence and responsibility of the input data to the output results.

Regarding the second conclusion, most approaches handle the problem of quantifying the influence and responsibility of the input data to the output results by finding all possible valuations of the rest of the input, for which an input under consideration directly determines the output value. Then, they calculate the total probability for these valuations to hold, and this is the influence/responsibility of the input under consideration.

Chapter 3

Pipeline Model

3.1 Model Description

Our system consists of a set of uncertain operators. Composing multiple operators together gives us a plan, i.e., a hierarchical pipeline. We assume that the input data of the pipeline (base input) is uncertain. The results are produced through the execution of a plan, which in turn consists of executions of each individual operator in the pipeline.

We introduce a directed acyclic graph *G=(V,E)*, for representing the (extraction and integration) hierarchical pipeline, i.e., the topology and correlations between operators. Following [5], this can also be seen as a *causal network*. V denotes the set of vertices corresponding to the operators (from now on we will use the terms node and operator interchangeably as each node refers to one operator), and E is the set of edges, showing how operators are connected to each other. An edge $v_i \rightarrow v_j$ between nodes v_i and v_j indicates that the output of the operator represented by node v_i is input to the operator represented by node v_j. Thus, edges E indicate each node's parents and children. By parents of a child node, we denote all the nodes that feed directly the child node with input, so v_i is parent of v_j and v_j is child of v_i.

We often use the language of genealogy in order to describe the relationships in the graph, for simplistic reasons.

We call nodes without parents *leaf nodes*, and the rest *intermediate nodes*. *Root node* is the top-level node of the system, which has no children, and provides

the system's output.

Also, we assume an ordering I (topological order) where node v_i can only be child of nodes v_j with j>i. Note that, since a pipeline is defined by a directed acyclic graph (DAG), there is such a topological ordering, i.e., an ordering of the vertices such that the starting endpoint of every edge occurs later in the ordering than the ending endpoint of the edge. Node v_1 is the root node.

3.2 Preliminaries

Before proceeding to the underlying theory and the presentation of our approach, we provide some useful definitions and summarise the notation that will be used in the ensuing discussion.

3.2.1 Definitions

Let path(v_i,v_j) denote a path from node v_i to node v_j:

path(v_i,v_j)=E(v_i,v_{k_1})+E(v_{k_1},v_{k_2})+...+E(v_{k_m},v_j).

We denote with D_i the set of all descendant nodes of a node v_i:

$v_m \in D_i$ if $\exists$ path(v_i,v_m)

Node v_i can only affect -with its output- all nodes $v_m \in D_i$. This corresponds to the *space of influence* of v_i.

We continue by defining the set C_i, which contains all non-descendant nodes of v_i that are parents of any descendant node of v_i, i.e., all other direct inputs of the descendants of v_i. More formally, we have:

$v_j \in C_i$ iff $\exists$ E(v_j,v_k), where $v_k \in D_i$ and $v_j \notin D_i$

Then, all nodes in C_i are the nodes that directly connect to v_i's space of influence.

The topology of the sets C_i and D_i for an example lineage DAG is depicted in Figure 3.1.

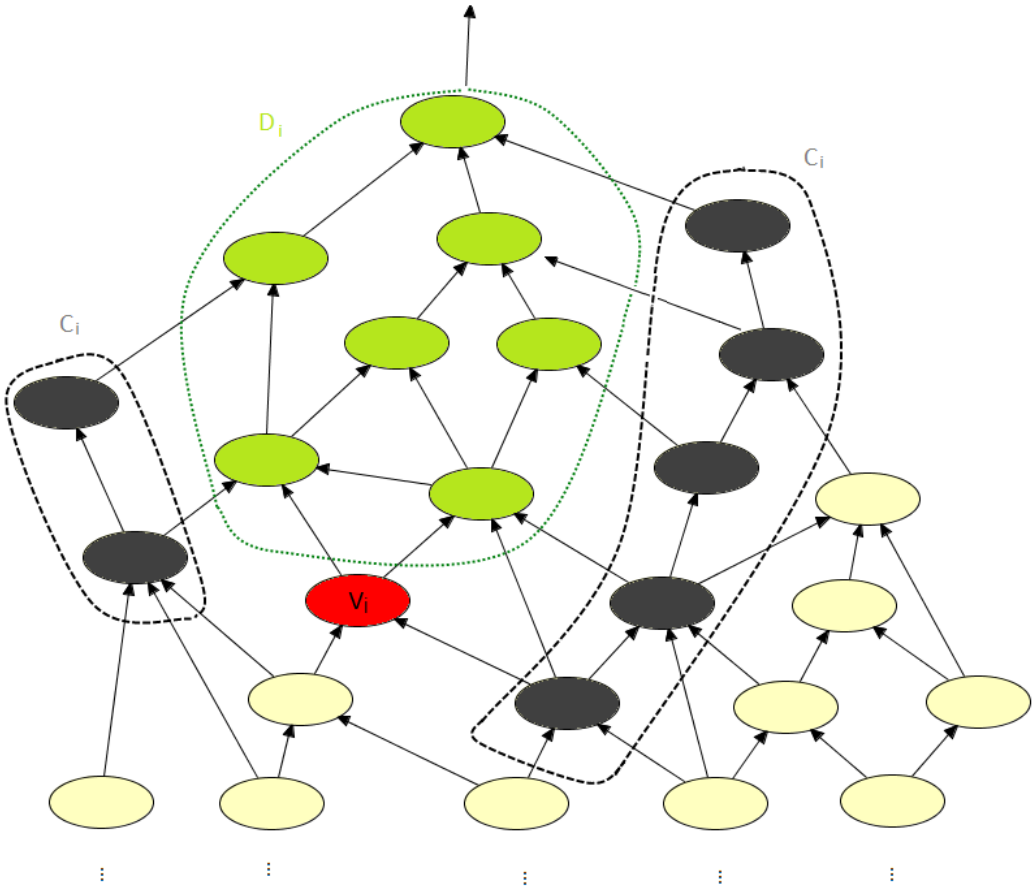


Figure 3.1: C_i and D_i sets for node v_i of a sample pipeline

Node v_i is coloured red, and the nodes of the sets C_i and D_i are coloured with gray and green, respectively.

The nodes of the set C_i can be seen as a "barrier" between the rest of the graph and the subgraph defined by node v_i and all its descendant nodes up to the root node(v_1). We use C_i to "isolate" D_i from the rest of the graph and focus on v_i's influence on its descendants and the root node. Because of the described characteristics of the pipelines that we handle, clearly $v_1 \in D_i$, for every node v_i.

We should point out that the nature of our model and the graph structure proposed prohibit us from assuming independence among nodes of the set C_i. The only independence assumption throughout our work is the independence among

the leaf nodes.

3.2.2 Notation

We denote the domain of a random variable Z by Dom_Z. Atomic events are propositions of the form $\mathrm{Z}_i = \mathrm{z}_i$ where Z_i is a random variable and $\mathrm{z}_i \in \mathrm{Dom}_{Z_i}$ is one of its domain values. An event is the assignment of all random variables to one of their domain values.

We define finite probability distributions via a set of random variables with finite domains. Such a probability distribution is completely specified by a function P that assigns a number $\mathrm{P}(\mathrm{Z}_i = \mathrm{z}_i) \in [0,1]$ to each atomic event $\mathrm{Z}_i = \mathrm{z}_i$ such that, for each random variable Z_i, $\sum_{z_i} \mathrm{P}(\mathrm{Z}_i = \mathrm{z}_i) = 1$.

The domain of any random variable Z_i that we use throughout our work is Dom_{Z_i}=[0,1]. We allow to assign zero or one probability at an output value, in order to also represent certain operators. As a result, the pipelines we handle may contain both certain and uncertain operators. The way we handle pipelines with both certain and uncertain operators will be explained in the next sections.

We continue by providing a summary of the notation that is used in the rest of this work. Detailed definitions for some symbols are provided later in this chapter.

Symbol	Description
G(V,E)	directed acyclic graph with V vertices (random variables/nodes/operators) and E edges (variable dependencies)
v_i	node $v_i \in V$
pa_i	node set of the parent nodes of node v_i
path(v_i,v_j)	a directed path from node v_i to node v_j
D_i	the set of all descendant nodes of a node v_i
C_i	the set of all nodes that directly connect to a descendant node of v_i
Z	random variable
Dom_Z	domain of random variable Z
Z=z	valuation of random variable Z, $z \in Dom_Z$
P(Z=z)$\in$[0,1]	random variable distribution, corresponding operator output probability
Y_i	random variable on node's v_i output value
y_i	(observed) output value of node v_i, $y_i \in Dom_Y$=[0,1]
$\overrightarrow{y}_{pa_i}$	(observed) output values of the parent nodes of v_i
X_i	random variable on the truth assignment of node v_i's output value
x_i	valuation of X_i, $x_i \in Dom_X$=[True,False]. X_i=True iff Y_i=y_i
$P_{fp}(v_i, \overrightarrow{y}_{pa_i})$, $P_{fn}(v_i, \overrightarrow{y}_{pa_i})$	probability that y_i is a false positive/negative
$\mathcal{PW}$	a set of possible worlds
pw^k	the k^{th} possible world, $pw^k \in \mathcal{PW}$
P(pw^k)	the probability of pw_k, i.e. the joint probability distribution of all random variables in pw_k
$\overrightarrow{y^k_{C_i}}$	k^{th} valuation of the nodes $\in C_i$
Vy_{C_i}	the set of all k possible valuations of nodes $v_j \in C_i$
$Vy^{DV}_{C_i}$	the set of all valuations of nodes $v_j \in C_i$, for which v_i is a deciding vote
V_{DV}	the set of the deciding vote nodes

3.3 Uncertainty representation

A natural way to deal with uncertain data is to interpret all uncertainties in the system as probabilistic values in the interval [0,1].

We assume a set of random variables which model the problem we study.

Since the uncertainty in our setting derives both from the input values and the intermediate operators output values, we use these random variables to describe both the input and the intermediate output values.

3.3.1 Probabilistic base input

We call *base input* the input data to the pipeline. This input data comes from an extraction task. As a result, the base input is inherently uncertain. For this reason, we represent base input with random variables, exactly as we do with the output of the uncertain operators, as explained in the next section.

We assume that the base input is stored in and provided by a probabilistic database.

3.3.2 Uncertain Operators

In statistical hypothesis testing, there are two types of incorrect conclusions or errors that can be drawn. If a hypothesis is incorrectly rejected, when it should in fact be accepted, it is called a false negative. A false positive occurs when a hypothesis is incorrectly accepted when it should in fact be rejected.

In our model we consider a set of uncertain *binary* operators. Operators correspond to basic units of functionality (e.g., web page classification). Given some input data, an uncertain operator may output either the correct or the incorrect value for this input, i.e., either a true or a false positive/negative value (from now on, by negative we will refer to value "0" and by positive to value "1"). Such uncertain operators form the aforementioned hierarchical pipeline, each represented by a vertex $v_i \in V$ in graph G.

Let Y, $Dom_Y=\{0,1\}$ be a random variable that describes the uncertain output value of each operator and y the observed output value. We call *observed output value* the output value y_i of the execution of an individual operator $v_i \in V$, on some input $\overrightarrow{y}_{pa_i}$, where $\overrightarrow{y}_{pa_i}$ the observed output values of the parent nodes of node v_i (v_i's inputs). An operator input can be either some base input or the output of some operators of a previous level. Each operator is also defined by an *operator Input-Output (IO) specification*, consisting of the input it consumes and

the output it produces. Thus, for some base input, a plan execution provides us with the observed output value y_i of every operator v_i.

During the training phase, using a training set, we run a series of executions in order to learn what the observed output value of a node is, for every possible input. This is feasible, as the output value of each operator is deterministic -the same output value will occur no matter how many times we try the same input. As a result, we consider the observed output value of an operator for a given input, known.

However, the credibility of this output value is under question, i.e., we are not sure if the observed output value is correct for the observed input, due to the aforementioned operator's inherent uncertainty. The output value uncertainty does not indicate that we do not know what the observed output value will be for some input, but rather a degree of uncertainty over its correctness. As a result, we assign a probability on every output value to be a true/false positive/negative: the probability distribution of Y reflects the confidence we put on each atomic event $Y = y$, based on the false positive/negative ratios. Such false positive/negative probabilities are often estimated during, eg., classifier learning, as shown below.

In Figure 3.2 we show an uncertain operator during the training phase. This explains better the procedure through which we gain all the needed information in order to describe an uncertain operator, i.e., identify the observed output value given some input and calculate its false positive/negative ratios. This specific operator is responsible for deciding whether some given inputs can be categorized as food. As a training set, we use a set of words that come from an extraction process.

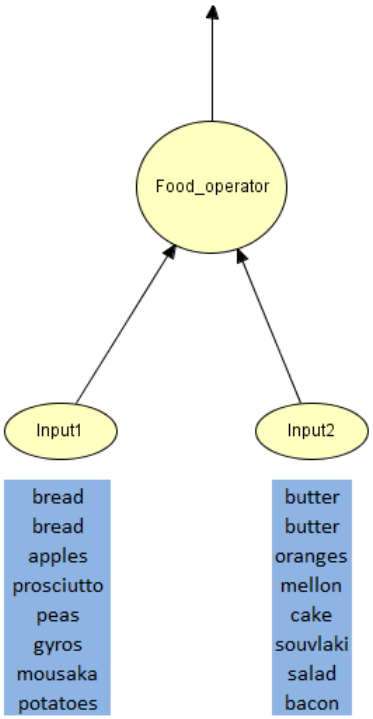


Figure 3.2: An uncertain operator given training data

We continue by providing in Figure 3.3 the training data used for training the operator of Figure 3.2.

	Sentence	Extracted words	
		W1	W2
S1	... as important as bread and butter ...	bread	butter
S2	... we also offer bread and butter before your lunch ...	bread	butter
S3	... it is like comparing apples to oranges ...	apples	oranges
S4	... where you can taste delicious prosciutto with mellon ...	prosciutto	mellon
S5	... the Black Eyed Peas where great. It was a piece of cake for them to ...	peas	cake
S6	... in Greece, where we tasted gyros and souvlaki ...	gyros	souvlaki
S7	... offering traditional mousaka and Greek salad ...	mousaka	salad
S8	... who disliked stuffed potatoes with bacon ...	potatoes	bacon

Figure 3.3: Example of training data used to estimate false positive/negative probabilities

In Figure 3.4 we present the input values and probabilities, the output values as resulted from an execution, and also the true/false positive/negative values

generated.

Sentence	Correct output	Probability of extracted words		Operator observed output	False positive/negative	
	Food (Yi={0,1})	P(W1)	P(W2)	Food (yi={0,1})	FP	FN
S1	0	0.95	0.95	1	Y	N
S2	1	0.95	0.95	1	N	N
S3	0	0.95	0.95	1	Y	N
S4	1	0.85	0.9	1	N	N
S5	0	0.9	0.95	1	Y	N
S6	1	0.4	0.25	0	N	Y
S7	1	0.75	0.95	1	N	N
S8	1	0.95	0.9	1	N	N

Figure 3.4: Sample operator's IO specification and output correctness

We observe that the system for the sentences S1, S3 and S5 generated a false positive value and for sentence S6 generated a false negative value. As a result, we may calculate that for this operator and this training data, its false positive ratio is 3/8=0.375 and its false negative ratio is 1/8=0.125.

In Section 3.4 we show that the introduced hierarchical pipeline model can be described using a Bayesian network. Thus, it fulfils the local Markov property. Also, as the graph is acyclic, we are guaranteed that a node cannot be dependent on its own value. The above indicate that every node's output value is conditionally independent from all other nodes output values apart from its parents, given $\overrightarrow{y}_{pa_i}$; the observed output value y_i just depends on $\overrightarrow{y}_{pa_i}$. In other words:

$Y_i \perp \overrightarrow{Y}_{anc_i} \mid \overrightarrow{y}_{pa_i}$, where $\overrightarrow{Y}_{anc_i}$ the variables for ancestor nodes of v_i in the pipeline, and $\perp$ denotes independence.

Note that, in our setting, the uncertainty of the output value of an operator is independent from the uncertainty of its inputs: the uncertainty of the output value depends only on its observed input values and the operator's false positive and false negative ratios for these input values. How uncertain the input of v_i is, does not affect how uncertain the output of v_i will be, only the observed input values $\overrightarrow{y}_{pa_i}$ determine y_i's uncertainty.

We denote as $P_{fp}(v_i, \overrightarrow{y}_{pa_i})$ the probability of the operator of a node v_i to output a false positive (fp) value (a false "1"). Equivalently for false negative (fn) value, $P_{fn}(v_i, \overrightarrow{y}_{pa_i})$.

Each node's output can be one of the following:

$$P_{tn}(v_i, \overrightarrow{y}_{pa_i}) = P(Y_i{=}0 \mid y_i{=}0, \overrightarrow{y}_{pa_i}) \qquad \textit{true negative}$$

$$P_{fn}(v_i, \overrightarrow{y}_{pa_i}) = P(Y_i{=}1 \mid y_i{=}0, \overrightarrow{y}_{pa_i}) \qquad \textit{false negative}$$

$$P_{tp}(v_i, \overrightarrow{y}_{pa_i}) = P(Y_i{=}1 \mid y_i{=}1, \overrightarrow{y}_{pa_i}) \qquad \textit{true positive}$$

$$P_{fp}(v_i, \overrightarrow{y}_{pa_i}) = P(Y_i{=}0 \mid y_i{=}1, \overrightarrow{y}_{pa_i}) \qquad \textit{false positive}$$

The above introduce the probability of the observed output being true/false positive/negative.

Going back to the operator of Figure 3.2, denoting it as the node v_i, we write:

$P_{fn}(v_i, \overrightarrow{y}_{pa_i}) = 0.125$

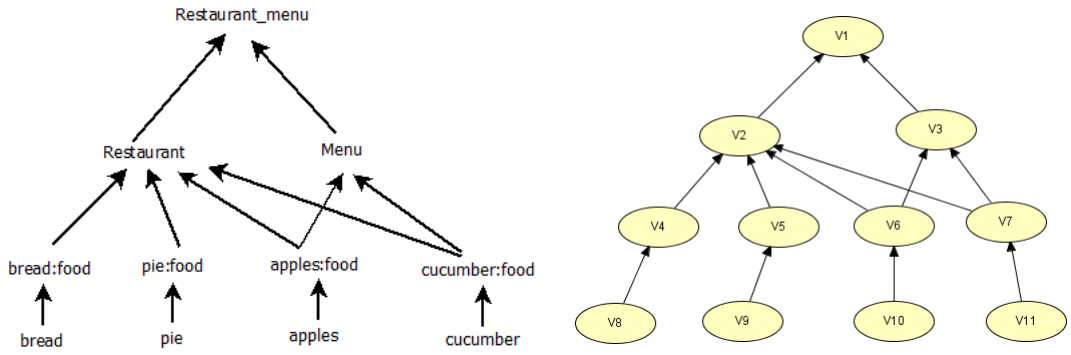
and

$P_{fp}(v_i, \overrightarrow{y}_{pa_i}) = 0.375$

Having studied thoroughly the uncertain operator as a unique entity, we should also examine how such operators interconnect to form the aforementioned pipelines.

Not only the operators add uncertainty to their output by producing false positives and false negatives, but also the input of an operator v_i can be the output of an uncertain parent operator v_j. Thus, operators always "operate" on uncertain inputs. There are two sources of uncertainty for each output: one deriving from the operator itself and another from its input, i.e., the confidence we put on the the output of the operators parent nodes.

Figure 3.5 shows a pipeline populated with uncertain operators.



(a) (b)

Figure 3.5: Restaurant menu classifier pipeline (a) and the corresponding pipeline (b) in graph representation

Figure 3.5 shows an extraction pipeline that can be described by our model, and is used to categorize web pages into restaurant menu pages. The pipeline's input data is keywords that come from a web extraction task. The intermediate operators categorize this data (e.g., Restaurant, Menu operators can be decision tree classifiers) by aggregating their results into more general categories, and the root node outputs the final result which declares the pipeline's decision on a web page being a restaurant menu page. Following the aforementioned common learning technique, we are provided with the false positive/negative ratios for all pipeline uncertain operators.

In Figure 3.6, we present the restaurant menu pipeline of Figure 3.5 in its full extent, including observed output values and probabilities, i.e., the result of a plan execution for some base input (here: bread, pie, apples, oranges).

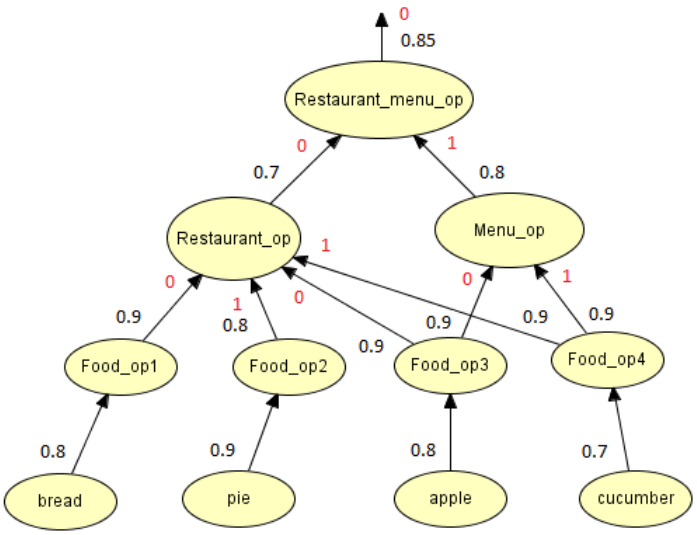


Figure 3.6: Restaurant menu classifier pipeline including the operators the observed output values and their corresponding probabilities

The probabilities represent the true positive/negative probabilities of the operators observed output values. Here, again, we show that the false positive/negative probabilities of all operators are independent from the input probabilities; they only depend on the observed input values to the corresponding operator.

We continue by introducing a Boolean random variable X, Dom_X={T,F} to denote the truth assignment on the correctness of an atomic event Y=y. Then, X_i=True $\equiv$ Y_i=y_i, and P(X_i=True) = P(Y_i=y_i | $\overrightarrow{y}_{pa_i}$) is the probability that the operator's observed output value is correct, given $\overrightarrow{y}_{pa_i}$. This variable will help us provide some useful definitions later in this chapter.

So far, we conclude with the following: the output value of each node is determined by a) the node's input values, b) the operator specifications and c) the uncertainty introduced by the node itself, due to false positives/negatives.

3.4 Bayesian Network

A Bayesian network consists of a directed acyclic graph (DAG) and a set of local distributions. Each node in the graph represents a random variable. A random variable denotes an attribute, feature, or hypothesis about which we may be uncertain. Each random variable has a set of mutually exclusive and collectively exhaustive possible values. That is, exactly one of the possible values is or will be the actual value, and we are uncertain about which one it is. The graph represents direct qualitative dependence relationships, and the local distributions represent quantitative information about the strength of those dependencies. The graph and the local distributions together represent a joint distribution over the random variables denoted by the nodes of the graph.

Up until now, it seems that the proposed model can be represented by a Bayesian network, as it has the following characteristics[1]:

- A set of random variables, i.e., the random variables Y_i, makes up the nodes of the network, which represent the uncertain operators.
- A set of directed links connects pairs of nodes. The intuitive meaning of an edge from node v_i to node v_j is that v_i has a direct influence on v_j.
- Each node has a conditional probability table (CPT) that quantifies the effects that the parents -partially, due to operator's inherent uncertainty- have on the node. The parents of a node v_i are all those nodes that have edges pointing to v_i.
- We assume that the graph has no cycles and all edges are directed (hence is a directed acyclic graph, or DAG).

This conclusion -that we may use a Bayesian network as a representational tool for our hierarchical pipeline model- gives as a very powerful background, as there is rich literature on Bayesian Networks, which will equip us with the appropriate tools to handle some aspects of this problem efficiently.

[1] We should point out that in our setting, the observed inputs of the pipeline determine the observations and the false positive/negative probabilities of each operator's output. As a result, it is not completely true to say that our model can be described by a Bayesian network

3.5 Possible Worlds

We use the standard possible worlds semantics from probabilistic databases to express all possible truth assignments of a set of operators (random variables).

A truth assignment $\overrightarrow{x}$=$\{x_1,\ldots,x_n\}$ of n random variables, representing our belief for each operator(node) v_i, i≤n, on Y_i=y_i is an assignment of each random variable X_i to one of its domain values. Every complete assignment maps to a single possible world.

Each possible world is described by: i) a positive integer(key) k of the possible world, ii) a truth assignment $\overrightarrow{x}$ on random variables and iii) the probability of the possible world. Then, $pw^k \in \mathcal{PW}$ is the k^{th} possible world, represented by the triplet:

<k,$\overrightarrow{x}^k$,P(pw^k)>

As each possible world represents a truth assignment on the output values of some (or all) nodes, the set of possible worlds $\mathcal{PW}$ covers all possible states of the set of operators under consideration. Of course, if |x|=n, then $1 \leq k \leq 2^n$ for binary output values.

Our assumption that most or all graph nodes are uncertain, i.e., $P_{fp}(v_i, \overrightarrow{y}_{pa_i}) \geq 0$ and $P_{fn}(v_i, \overrightarrow{y}_{pa_i}) \geq 0$, $\forall\ v_i$, indicates that each uncertain node's observed output can always be characterized as True/False. So, if there are n nodes (operators) in the DAG, there are at most 2^n possible worlds that describe all possible truth assignments.

We should point out that if there were no intermediate uncertain operators and the pipeline's uncertainty derived only from the base input -as it is in typical probabilistic databases, e.g.[1]- then the number of possible worlds would be much smaller, that is 2^m, where m the number of base inputs (leaf nodes).

As every node in the graph is conditionally independent to all other nodes except for its parents, given its parents output values, the probability of every possible world describing the root's output confidence is the joint probability distribution of all assignments X_i=x_i in $pw^k \in \mathcal{PW}$:

$$P(pw^k)=P(X_1=x_1^k,\ldots,X_n=x_n^k) = \prod_n P(X_i=x_i^k \mid \overrightarrow{y}_{pa_i}^k)$$

where $\overrightarrow{y}^{k}_{pa_i}$ denotes the observed output values of the parents of node v_i, i.e., the input values of v_i that we consider at the k^{th} possible world.

3.6 Lineage

Providing the lineage of a result in our setting is more complicated and challenging than in a common probabilistic database setting: following the typical possible worlds approach to describe a pipeline result's lineage, (i.e., find all its possible derivations based on the input values only) would be insufficient in our setting. In a typical probabilistic database setting, only the leaf nodes take part in the decision process and intermediate operators only propagate the information, they cannot interfere with it. We not only have to take into consideration input uncertainty, we should also assign a level of "blame", or responsibility, to each of the operators for an erroneous output.

Suppose that we are provided with an output value, a probabilistic database as input source and a pipeline populated with uncertain operators. The lineage function of an output results by taking into account all input and intermediate operators output combinations that may result in the observed output value under consideration.

However the overhead we get by calculating possible worlds in this new setting, the procedure remains quite the same: the lineage of an output value is a Boolean formula which evaluates to True for every assignment of all pipeline nodes, for which the result under consideration appears in the output. Informally, we apply to the whole graph what probabilistic database approaches do for the inputs only. In our setting, all graph nodes take part in the decision process and as a result we cannot ignore any of them, in order to keep the procedure lossless.

Chapter 4

Our approach

Our ultimate goal is to invest the least effort, time and resources to locate and refine a small subset of operators in order to gain the maximum benefit, in terms of the root operator's output quality. Informally, we intend to locate and rank the influential operators of a pipeline in order to be able to repair the pipeline with the least cost through refining some significant operators. Then, these refined operators are enough to boost the quality of the pipeline's result, without having to refine all pipeline's operators.

We approach this problem by defining each node's influence to the result and determine whether each pipeline operator is influential to the output or not.

4.1 Influence

We are interested in measuring the influence of each operator to the root node v_1. Among all graph nodes, there are some which, due to their topological characteristics and their output value, have greater influence to the root's output value and probability. Such nodes are called influential and are of great importance when aiming to improve the system with the least cost.

The term "influential node" may be confusing though; all graph nodes are responsible for the root's output as they all take part in the decision process. However, only some of them have the ability to directly determine the root's output value. These play the role of the so called deciding votes, which will be

better explained afterwards. Here, we use the term "influential node" to refer to the nodes that directly determine the root's output value.

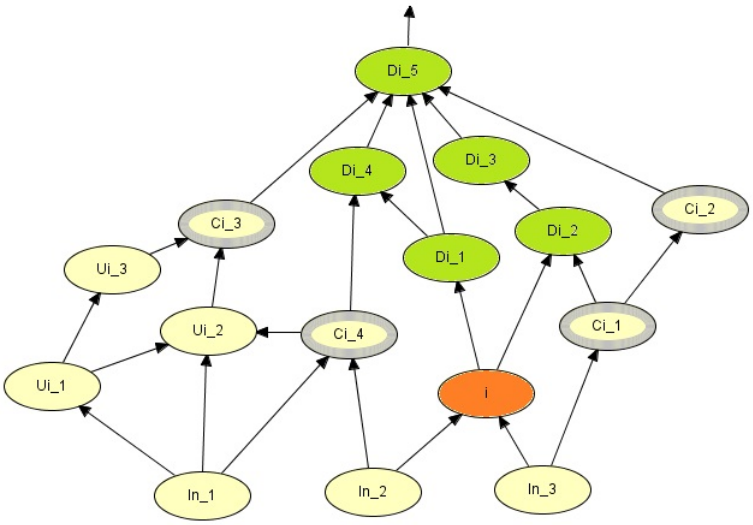
Unlike the classic definition of influence in probabilistic databases literature [1], where the influence is only credited to the independent leaf nodes, here any leaf or intermediate uncertain node can atomically influence the root's output, as most or all graph nodes are sources of uncertainty. Thus, we should examine every single uncertain operator for its contribution to the result's uncertainty. In our approach, the probability of the output result is not simply the joint probability of the corresponding input tuples; the probability of an output result comes from the probability of the root node for the corresponding uncertain input values. Recursively, every single root node output depends on the whole pipeline.

We continue our description, assuming that all pipeline operators output uncertain values and we will then discuss how we handle cases where some operators produce certain output. We consider this assumption because a certain operator is a sub-case of an uncertain operator, where one output value has probability 1 and the rest 0.

Let a node v_i be either a leaf or an intermediate node. The set of nodes C_i (see Section 3.2.1) forms a "slice" of the graph breadth-wise, a barrier which contains all nodes that separate the rest of the graph from the nodes which v_i influences (D_i). Nodes $\in C_i$ are the only nodes that also affect D_i nodes, apart from v_i.

The properties of the set C_i of a node v_i under consideration are an important tool and reduce our effort on finding v_i's influence to the root node, because we only have to work on the nodes that are in, or descendants of, the node set C_i. As a result, we cut down on the number of pipeline operators that we examine each time we calculate some node's influence.

The topology of the aforementioned node sets can be better understood through the following figure:



Green: D_i
Grey: C_i
Orange: v_i

Figure 4.1: Example of a pipeline

We should distinguish here two cases, according to the information we have about the pipeline:

4.1.1 Case I: Observed values

The input and output values of v_i and C_i nodes may be observed, as the outcome of an execution. In this case, we know the probability of their outputs, as it comes from each node's false positive/negative probability for the given input values.

The fact that the output values of C_i nodes are probabilistic suggests that we should consider all possible truth assignments of these output values, given their observed input values. Following the possible worlds semantics, as explained in Section 3.5, there are $2^{|C_i|}$ possible truth assignments on the observed values $\overrightarrow{y}_{C_i}$. A truth assignment, e.g., x_j^k=True indicates that in possible world k, we assume that the observed output value y_j of node $v_j \in C_i$ is not a false positive/negative.

We map each truth assignment k to a new valuation $\overrightarrow{y_{C_i}^k}$. This valuation can be interpreted as a "virtual" observation, where we ask "what if the k_{th} truth assignment holds". We consider the following:

- if x_j^k = True, then $y_j^k = y_j$

- if x_j^k = False, then $y_j^k = \neg y_j$

In simple words, what we do to create each valuation $\overrightarrow{y_{C_i}^k}$ is consider the k^{th} truth assignment, and adopt the corresponding output values.

Each possible world $pw^k \in \mathcal{PW}$ assigns a probability $P(Y_j{=}y_j^k \mid \overrightarrow{y_{pa_j}})$, $\forall\ v_j \in C_i$. Then, it is:

$$\sum_{k=1}^{2^{|C_i|}} P(pw^k) = \sum_{k=1}^{2^{|C_i|}} \prod_{j:v_j \in C_i} P(\overrightarrow{Y_j}{=}\overrightarrow{y_j^k} \mid \overrightarrow{y_{pa_j}}) = 1 \qquad (1)$$

4.1.2 Case II: Non-observed values (General case)

If no observed input and output values of C_i nodes are provided, we have to take into account all possible valuations of C_i nodes. Then the set of truth assignments on observed values, where each truth assignment maps to a new valuation, is replaced by directly considering all possible valuations of the output values of C_i nodes. Each such valuation of all C_i nodes is represented by a possible world, whose probability is the joint probability of all valuation's output values to hold. This is provided by the operators CPTs: we compute the unconditional probability that the "assumed" observed value is correct, given the node's false positive/negative probabilities, weighted over all possible inputs.

4.1.3 Influence quantification

Following the probabilistic database approach, where we have to consider all possible worlds because we do not know what is the "real" state of the database, similarly here we consider all possible worlds to evaluate all possible "correct" states of v_i and its C_i set's nodes. Each possible world over the set C_i provides the probability of the respective valuation to hold. The possible worlds are either based on all possible truth assignments of an observed valuation, or on all possible valuations of a set of nodes (v_i,C_i) when no observation is available.

4.1.3.1 Valuation of sets C_i and D_i

As we have already shown in Section 3.3.2, through operators' I/O specifications, we know what the observed output of a node will be, given its observed input values. Thus, given the k^{th} valuation (k^{th} possible world) of all nodes in C_i and an output value for v_i, $(\overrightarrow{y^k_{C_i}}, y^k_i)$, the corresponding observed output values of the descendants of v_i, $\overrightarrow{y^k_{D_i}}$, are known. Therefore, every valuation $(\overrightarrow{y^k_{C_i}}, y^k_i)$ maps to a valuation $\overrightarrow{y^k_{D_i}}$, since the output values $(\overrightarrow{y^k_{C_i}}, y^k_i)$ determine what $\overrightarrow{y^k_{D_i}}$ will be, $\forall$ k.

However, as the nodes $v_m \in D_i$ are (also) uncertain operators, there is a probability on their output value to be correct for the corresponding input, given their observed input values. After considering an output value (y^k_i) for v_i for which we will calculate its influence to the root's output, the joint probability of the corresponding valuation of the dependent D_i nodes, is:

$$P(\overrightarrow{Y_{D_i}}=\overrightarrow{y^k_{D_i}}) = \prod_{m:v_m \in D_i} P(Y_m=y^k_m \mid \overrightarrow{y^k_{pa_m}}) \qquad (2)$$

which is the joint probability of the values we would observe in the subgraph at the k^{th} valuation, given $(\overrightarrow{y^k_{C_i}}, y^k_i)$. Note that, of course, not all nodes in D_i are directly fed with input by v_i or $v_j \in C_i$ nodes; the input of some nodes comes from another node $v_m \in D_i$. However, the "primary" input was provided by either v_i or $v_j \in C_i$.

Concluding, for each single valuation $\overrightarrow{y_{C_i}}$, there are $2^{|C_i|}$ possible truth assignments, each of which maps to a single virtual valuation of C_i nodes, $\overrightarrow{y^k_{C_i}}$. Each $\overrightarrow{y^k_{C_i}}$ also corresponds to a single valuation of D_i nodes, $\overrightarrow{y^k_{D_i}}$, given the output value of v_i.

4.1.3.2 Deciding votes

We remind that one of our ultimate goals is to measure how each uncertain operator influences the credibility of the root's output, i.e., the probability of it being correct. For this reason, we introduce the notion of deciding votes, as it appears in the causality literature [2].

We begin by examining, out of all $2^{|C_i|}$ possible valuations $\overrightarrow{y_{C_i}}$, which are the ones that make y_i the deciding vote.

Let $\lambda_{y_1}(\overrightarrow{y^k_{C_i}},\mathrm{y}^k_i) = \{ \begin{array}{ll} 1 & \text{, if for given } (\overrightarrow{y^k_{C_i}},\mathrm{y}^k_i)\text{: } \mathrm{y}^k_1 = 1 \\ 0 & \text{, if for given } (\overrightarrow{y^k_{C_i}},\mathrm{y}^k_i)\text{: } \mathrm{y}^k_1 = 0 \end{array}$

Then, checking:

$[\lambda_{y_1}(\overrightarrow{y^k_{C_i}},\mathrm{y}^k_i) \neq \lambda_{y_1}(\overrightarrow{y^k_{C_i}},\neg\mathrm{y}^k_i)]$ (3)

returns the valuations $\overrightarrow{y^k_{C_i}}$ for which y^k_i is the deciding vote for y_1, based on observed values. That is, for which out of all possible valuations $\overrightarrow{y^k_{C_i}}$, the output value of v_i determines what the root's output value will be. Informally, we examine the effect of how v_i's output value changes the root's output value.

We write:

$\mathrm{v}_i \in \mathrm{V}_{DV}$ if the deciding vote criterion (3) holds.

If Vy_{C_i} denotes the set of all $2^{|C_i|}$ possible valuations $\overrightarrow{y_{C_i}}$, then $\mathrm{Vy}^{DV}_{C_i} \subseteq \mathrm{Vy}_{C_i}$ denotes the set of all valuations for which v_i is the deciding vote ($\mathrm{v}_i \in \mathrm{V}_{DV}$); that is, $\mathrm{Vy}^{DV}_{C_i}$ contains every $\overrightarrow{y^k_{C_i}}$ for which (3) holds, or equivalently the corresponding valuations of these possible worlds. In simple words, $\mathrm{Vy}^{DV}_{C_i}$ contains all valuations of nodes in C_i, as they resulted from the corresponding truth assignments, for which a change in v_i's output value results in the change of v_1's output value.

The above property of the valuations $\mathrm{Vy}^{DV}_{C_i}$ can well be seen as the *support* under which v_i becomes the deciding vote. If $|\mathrm{C}_i|$=0 and (3) holds, or $\mathrm{Vy}_{C_i} \equiv \mathrm{Vy}^{DV}_{C_i}$, then v_i is always the deciding vote.

Having examined the supporting role of C_i nodes, we should also examine the role of D_i nodes in whether v_i is the deciding vote in each valuation. D_i nodes practically propagate the decision of the node v_i to the root node. We distinguish two methods to examine the role of the set D_i:

4.1.3.3 All D_is

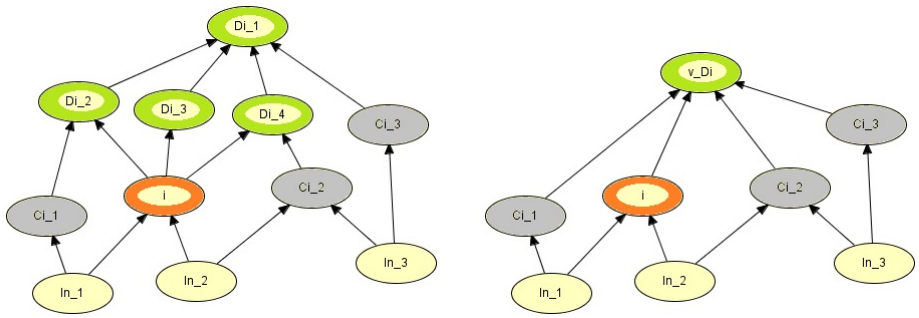
For all valuations $\mathrm{Vy}^{DV}_{C_i}$, we choose to ask for all descendant nodes of v_i to be assigned True, in order to better capture the influence of v_i to the root node. As each node's output uncertainty depends on its input values, we have to ensure that this input is as correct as possible.

We can calculate the total probability that y_i is a deciding vote:

$$\sum_{k=1}^{|Vy^{DV}_{C_i}|} \prod_{j:v_j \in C_i \cup D_i} \mathrm{P}(\mathrm{Y}_j{=}\mathrm{y}^k_j \mid \overrightarrow{y_{pa_j}}) \qquad (4)$$

Considering the joint probability of all D_i nodes, $\overrightarrow{y^k_{D_i}}$, for every possible $\overrightarrow{y^k_{C_i}}$ is basically equivalent to reducing all descendants of v_i in to a single node v_{D_i}, whose inputs are v_i and v_{C_i} nodes, and calculating its output value probability. Asking for the probability that the output of all D_i nodes is True for the corresponding $\overrightarrow{y}^k_{C_i}$ is as if we ask for v_{D_i} to be True under this new setting.

The above can be better understood through the following figure:



(a) (b)

Figure 4.2: Reduction of descendant nodes D_i (a) into a single node (b).

Following this approach, we have the following definition of influence:

Definition: The influence of the observed output value y_i of a node v_i to the observed output value y_1 of the root node v_1 is defined as the sum of the joint probabilities of all subgraph $(C_i \cup D_i)$ valuations on being true positives/negatives, for which y_i is the deciding vote for y_1.

We write:

$$\boxed{\mathrm{Inf}_{y_i}(y_1) = \sum_{k=1}^{|Vy^{DV}_{C_i}|} \prod_{j:v_j \in C_i \cup D_i} P(Y_j{=}y^k_j \mid \overrightarrow{y_{pa_j}})} \qquad (5)$$

The definition of the influence of a node v_i to the root node (v_1) calculates the total probability of all nodes $\in C_i \cup D_i$ having "correct" values, for which v_i's output value is the deciding vote for v_1's output value. Informally, it answers the question: "What is the probability that y_i will -correctly- determine the output? How much do we trust the hypotheses of the truth assignments of all other nodes, for which v_i determines the result?"

4.1.3.4 Root-only

Another method to define y_i's influence is to ignore the output probabilities of all D_i nodes and only examine the root node's output probability, for all valuations $Vy_{C_i}^{DV}$. We weight each output probability with the probability of the respective possible world, from which it has emerged.

We write:

$$\mathrm{Inf}_{y_i}(y_1) = \sum_{k:pw^k \in Vy_{C_i}^{DV}} \mathrm{P}(Y_1{=}y_1^k \mid \overrightarrow{y_{pa_j}}) \times \mathrm{P}(pw^k) \qquad (6)$$

We describe this naive but "cheap" method in order to compare it to the other two and argue about the trade-off between the computational cost and the quality of the results.

4.1.3.5 Total influence

As we can see, the described approaches require to apply an output value to the node under consideration, v_i. Therefore, measuring the influence by taking into account this valuation will refer to the influence of the *particular* output value of v_i to the root node. However, we intend to measure the influence of the operator, not the influence of a specific output value of it.

Thus, we have to examine both possible output values of the node v_i in order to compute the influence of v_i to v_1. Then, we weight over the probability of each possible output value to determine the total influence.

We write:

$\mathrm{Inf}_{v_i}(v_1) = \sum_{b=0}^{1} \mathrm{Inf}_{y_i^b}(y_1) \times \Phi(b)$, where

$$y_i^b = \begin{cases} y_i & \text{, if b=0} \\ \neg y_i & \text{, if b=1} \end{cases} \quad \text{and} \quad \Phi(b) = \begin{cases} P(Y_i = y_i) & \text{, if b=0} \\ P(Y_i = \neg y_i) & \text{, if b=1} \end{cases}$$

4.1.4 Summary of our approach, in steps

We summarize the above procedure in the following steps:

1. We consider the observed output values of all v_{C_i} nodes and v_i and we form the set of all possible worlds describing all possible truth assignments over

the observed output values $\overrightarrow{y}_{C_i}$. If no observation is available, we consider directly all possible valuations of C_i nodes.

2. We map each possible world k to the corresponding virtual valuation $\overrightarrow{y}^k_{C_i}$.
3. For each valuation k and for y_i, we calculate the observed output values $\overrightarrow{y}^k_{D_i}$.
4. We identify for which of these valuations is y_i the deciding vote for y_1.
5. We sum over all possible worlds where the deciding vote criterion holds.

This is the influence of v_i to v_1 for the current observation.

4.1.5 Generalization of Re and Suciu [1] definition of influence

A very important property of the above definition of influence is that it can well be seen as a generalization of the influence definition provided by Re and Suciu in [1], extended to handle uncertain operators. Our definition reduces to the definition of influence provided by [1], when considering the case where the intermediate operators add no uncertainty to their output; all system's uncertainty derives from the independent leaf nodes. The reduction of our method to the definition of influence, as described in Re and Suciu paper [1], is quite straightforward.

Suppose that we examine such a simple setting, where no intermediate node adds further uncertainty to its output. As a result, all system's uncertainty derives from the leaf nodes, i.e., the probabilistic base input of the system. Therefore, we need only examine the leaf nodes for measuring each single node's influence to the result.

Let v_i be a leaf node and all intermediate pipeline operators be certain. Then the set C_i contains all other leaf nodes except for v_i. This is because intermediate nodes add no uncertainty and thus the nodes connecting to v_i's space of influence (nodes in the set C_i), reduce to their ancestor leaf nodes.

As a result:

$$P(\overrightarrow{Y_{C_i}}=\overrightarrow{y^k_{C_i}}) = \prod_{m=1}^{|C_i|} P(Y_{C_i,m}=y^k_{C_i,m}) = \prod_{|leaf\_nodes\setminus\{v_i\}|} P(\overrightarrow{Y_{leaf}}=\overrightarrow{y^k_{leaf}}) \quad (8)$$

assuming that leaf nodes are independent. Equation (8) holds, since leaf nodes are independent and we are provided with the unconditional probability of their output.

Using (4) we are able to calculate the joint probability of all other leaf nodes (C_i) to output such values, that v_i becomes the deciding vote. Then, we sum over all input value combinations' joint probabilities for which v_i is the deciding vote.

We have:

$$(4) \overset{(8)}{=>} \sum_{k=1}^{|V^{DV}_{leaf\_nodes \setminus v_i}|} \prod_{j=1}^{|leaf\_nodes|-1} P(Y_j{=}y_j^k) \qquad (9)$$

Moreover, the definition of influence as it appears in [1] is as follows:

$\text{Inf}_{x_i}(\lambda_t) \overset{\text{def}}{=} \mathbf{P}[\lambda_t(A) \neq \lambda_t(A \oplus \{i\})]$

The above indicates the sum of the probabilities of possible worlds for which the leaf node x_i is the deciding vote, i.e., toggling x_i's value toggles output value, and this is shown through the lineage function.

The above definition can be expressed by using possible worlds semantics as follows:

It is: $\text{Inf}_{x_i}(\lambda_t) \equiv \text{Inf}_{x_i}(q)$, where q the query result, and

$$\text{Inf}_{x_i}(q) = \sum_{k:pw_i^k \in V^{DV}_{leaves \setminus x_i}} \prod_{j:x_j \in \overrightarrow{x} \setminus x_i} P(Y_j{=}y_j^k) \qquad (10)$$

where $\overrightarrow{x}$ the set of all leaf nodes and $V^{DV}_{leaves \setminus x_i}$ the set of valuations (possible worlds) over all leaf nodes except for x_i, for which x_i is the deciding vote.

As a result, equation (9) agrees with equation (10), proving that our definition is a generalization of the definition of influence by Re and Suciu.

4.1.6 Influence desiderata

Intuitively, we consider that the higher the out-degree or the closest a node is to the root node, the higher is its influence.

In order to prove that our approach satisfies our intuition, we continue by defining some topological metrics that characterize the expected amount of influence of a node v_i.

We prove that our approach satisfies the following:

4.1.6.1 Node's out-degree

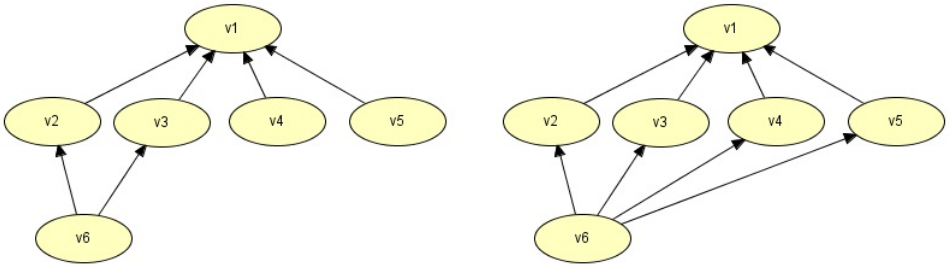
The out-degree of a node v_i ($\deg^+(v_i)$) is defined as the number of outgoing edges of v_i.

We write:

if $\deg^+(v_i) > \deg^+(v_j)$ then we expect $\text{Inf}(v_i) > \text{Inf}(v_j)$

The greater a node's out-degree ($deg^+(v_i)$), the more influential we expect it to be. This happens because toggling the output value of a node with high out-degree, this change "spreads" in a larger part of the graph, and thus it is more likely to cause a change to its descendants.

Example:



(a) (b)

Figure 4.3: Node out-degree

We will now examine the cases presented in Fig.3. Both cases (a,b) are almost the same; the only difference is the out-degree of node v_6. It is $deg^+(v_6)$=2 in (a) and $deg^+(v_6)$=4 in (b). In both cases, the distance of v_6 from the root is the same, dist_{v_6}=2 and the in-degree of the root node is $deg^-(v_1)$=4.

Let us begin by examining (a). There might be some valuations of C_6 nodes, C_6={v_4, v_5}, for which v_6 becomes a deciding vote. However these valuations may never occur, or they may have very low probability to occur. As a result, whether v_6 will become the deciding vote depends heavily on v_4 and v_5's output values.

Now let us take a look what changes in case (b). Changing $deg^+(v_6)$ from 2 to 4 means that we will now consider the nodes v_4 and v_5 from being in C_6, to be in D_6. They have now been "absorbed" by D_6 and have become descendants of v_6.

As a result, v_6's output value will determine what v_4 and v_5's output values will be. In (b), v_6 influences more graph nodes and so, it has the ability to provide larger part of the graph with its output.

Formally, the influence of v_6 in (a) is:

$$\mathrm{Inf}_{v_6}^{(a)}(v_1) = \sum_{k=1}^{|Vy_{C_6}^{DV}|} \prod_{j=1}^{|C_6|+|D_6|} P(Y_j{=}y_j^k) \qquad (11)$$

which calculates the sum of the joint probability of C_6 and D_6 nodes, for which v_6 becomes the deciding vote. However, if we calculate the influence of v_6 in (b), we get:

$$\mathrm{Inf}_{v_6}^{(b)}(v_1) = \sum_{k=1}^{|Vy_{C_6}^{DV}|} \prod_{j=1}^{|C_6|+|D_6|} P(Y_j{=}y_j^k) = \sum_{k=1}^{|Vy_{v_6}^{DV}|} \prod_{j=1}^{|D_6|} P(Y_j{=}y_j^k) \qquad (12)$$

If we take a closer look at the above equations we can see that, at the sum function of (11), it is C_6 that "decides" whether v_6 will be a deciding vote ($Vy_{C_6}^{DV}$). In (12) there is no C set, as its nodes have become descendants of v_i. Now v_6's output value feeds all nodes that directly connect to the root node and that, it is the one that determines their output values.

In fact, if we denote as $\mathrm{Inf}^{(a)}$ and $\mathrm{Inf}^{(b)}$ the influence of v_6 in cases (a) and (b) respectively, we have:

$\mathrm{Inf}^{(a)} \leq \mathrm{Inf}^{(b)}$, and the equality holds if v_6 in case (b) is a deciding vote for every valuation of C_6 nodes (v_4 and v_5).

The high influence and significance of a node with high out-degree seems more logical, if we consider a case where such a node outputs an erroneous value. This will flood the pipeline with error and result in probably erroneous root output.

4.1.6.2 Distance

The distance of a node v_i from another node v_j is measured by the number of nodes that lay between v_i and v_j. However, we have represented the pipeline with a directed acyclic graph, meaning that there might be many paths through which we can approach v_j from v_i. For this reason we take the shortest path (geodesic distance).

We write:

if $\mathrm{dist}(v_{i_1},v_j) > \mathrm{dist}(v_{i_2},v_j)$ then we expect that $\mathrm{Inf}(v_{i_1},v_j) < \mathrm{Inf}(v_{i_2},v_j)$

Practically, we expect that the farther a node v_i is from another node v_j, the less it influences v_j. As a result, a node close to the root node is expected to be more influential. This happens because high-level nodes are less hops away from the root node, and as the graph width narrows while moving up, a change in a higher level node is expected to have significant impact to the root.

This conclusion seems logical, as high level nodes (nodes close to the root node) tend to be fewer and carry more information (information aggregated by many nodes), so their significance is greater. Informally, it is the difference between the impact of a member of the board of directors to a strategical decision of an organization, and the impact of the member of a department. Generally, we may say that the less nodes are in v_i node's C_i and D_i sets, the more influential v_i is.

Example:

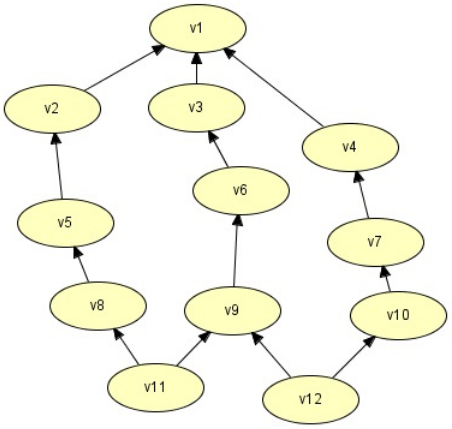


Figure 4.4: Nodes with different distances to the root node

Consider the case where we compare the influence of two nodes, v_2 and v_9 from Fig.2. We see that both have the same out-degree ($deg^+(v_2)$=$deg^+(v_9)$=1), however v_2 is much closer to the root than v_9. In fact, v_2 is directly connected to the root node, where v_9 is 3 hops away. Also, both nodes have the same size of C set. The sets C and D for these nodes are:

$C_2 = \{v_3, v_4\}$

$D_2 = \{v_1\}$

$C_9 = \{v_2, v_4\}$

$D_9 = \{v_6, v_3, v_1\}$

For simplistic reasons, let us assume that both nodes are deciding votes for the current setting, for 50% of the valuations of their C_i sets.

It is:

$$\mathrm{Inf}_{v_2}(v_1) = \sum_{k=1}^{|Vy^{DV}_{C_2}|} \prod_{j=1}^{|C_2|+|D_2|} P(Y_j{=}y_j^k \mid \overrightarrow{y^k_{pa_j}}) = \sum_{k=1}^{2} \prod_{j=1}^{3} P(Y_j{=}y_j^k \mid \overrightarrow{y^k_{pa_j}})$$

and

$$\mathrm{Inf}_{v_9}(v_1) = \sum_{k=1}^{|Vy^{DV}_{C_9}|} \prod_{j=1}^{|C_9|+|D_9|} P(Y_j{=}y_j^k \mid \overrightarrow{y^k_{pa_j}}) = \sum_{k=1}^{2} \prod_{j=1}^{5} P(Y_j{=}y_j^k \mid \overrightarrow{y^k_{pa_j}})$$

Assuming the same probabilities for all C_i and D_i sets for both nodes, we have that:

$\mathrm{Inf}_{v_9}(v_1) < \mathrm{Inf}_{v_2}(v_1)$

practically because $|C_9|+|D_9|=5 > 3=|C_2|+|D_2|$

4.1.7 Incorporating and managing certain operators

As we mentioned earlier, we consider pipelines that totally consist of uncertain operators. However, our approach can also be applied to pipelines which consist of both certain and uncertain operators. The incorporation of certain operators is straightforward: a certain operator is described as an uncertain where all possible outputs for an input have probability equal to zero apart from a single output value which has probability equal to one. Certain operators in a mostly uncertain-operator populated pipeline play the role of a stepping stone towards the effort to improve the pipeline.

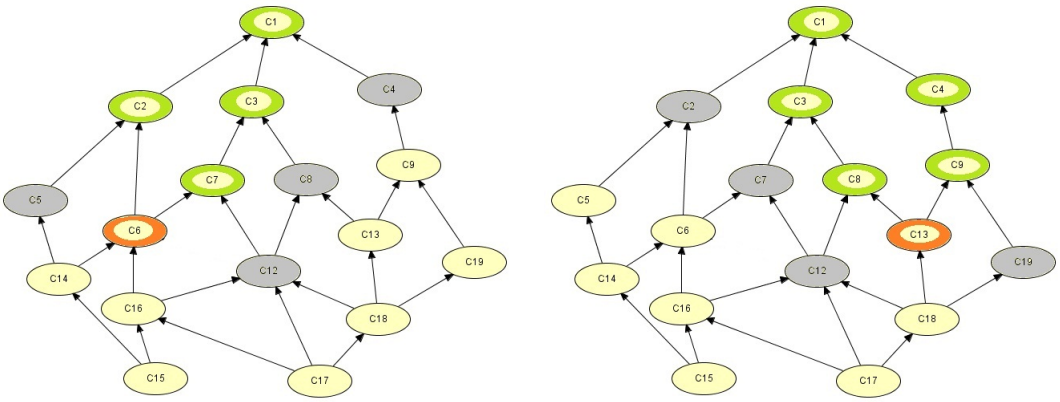
As a result, our approach can be applied to any pipeline, i.e., pipelines with i) only uncertain operators, ii) both certain and uncertain operators, and iii) only certain operators(in this case we have proven the reduction of our approach to the definition of influence in [1]).

The existence of a single certain operator will also cut down on the size of possible worlds, as it will assign zero probability to all possible worlds that do not meet the constraint set by the way the certain operator works, i.e., consider output values other than the one which is the correct/certain value for the respective

input. Informally, some possible worlds cannot exist (will have zero probability) as they will consider a zero probability input-output combination for the certain operator.

4.1.8 Measuring the influence of multiple nodes simultaneously

We extend our approach by considering the case where we want to calculate the "joint" influence of multiple operators simultaneously.



(a) (b)

Figure 4.5: C_i and D_i sets of nodes v_6 and v_{13} respectively

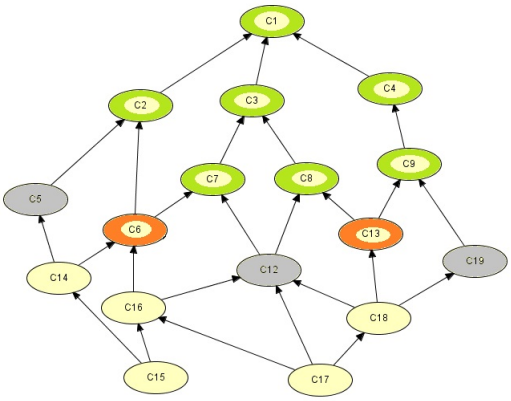


Figure 4.6: Sets $C_{6,13}$ and $D_{6,13}$

We will examine the joint influence of nodes v_i and v_j. The new C_{ij} and D_{ij} sets result from the intersection of the previous C_i, C_j and D_i, D_j sets respectively. Every possible output value combination (y_i,y_j) is considered as a new state, for which we calculate the influence of both v_i, v_j to the root's output, through C_{ij} and D_{ij}, just like we did when examined a single node.

However, we consider that the study and the challenges that arise when determining the influence of multiple nodes simultaneously are outside the scope of this thesis. We currently investigate the properties of joint influence with the intuition that although the high complexity of the problem, it will reveal a new facet of influence in uncertain operators pipelines. We leave the complete study of the influence of multiple nodes simultaneously for future work (Section 6.2).

4.1.9 Root node influence

At this point, we should examine how the influence metric works in case of the root node. Clearly, the presented metric cannot be applied to the root node, since there are neither C_i nor D_i sets for the root node.

However, we think that it is obvious that the root node operator is the operator that is responsible for the whole pipeline's output. As a result, should the root be erroneous, then consequently the final result would be wrong. Thus, there would be no point improving any other pipeline operator, if root node would ruin this effort.

This remark leads us to draw the conclusion that the root node v_1 is *always* the most influential node of any pipeline, and so, we say that: $\text{Inf}_{v_1} = 1$.

4.1.10 Calculating local influence

Another aspect of our approach is its ability to be used in order to perform an influence analysis at only a small part of a larger pipeline. For example, let us assume that we need to know the most influential nodes in case that the pipeline under consideration is a part/subgraph of a larger graph. Then, we may set a starting set of nodes and an ending node, as the leaf and the root nodes respectively and *locally* discover the top-k influential nodes.

Suppose that we are given a website classification pipeline populated with uncertain classifiers, whose task is to gradually classify a given website into a restaurant website. In such a pipeline there might exist food, wine, and dessert classifiers at the bottom levels of the pipeline, and at the upper levels we may find more complex classifiers, using the results of previous levels. We want to examine the influence of some intermediate food classifiers to an also intermediate restaurant menu classifier at an upper level. Such information can be immediately drawn by our approach, enabling local influence study of any pipeline.

4.1.11 Example

Now let us explain the above through an example. Suppose that we are given the following graph:

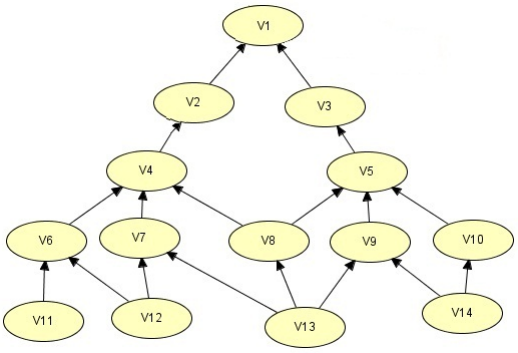


Figure 4.7: Example

We want to calculate the influence of v_7 to the output of v_1, supposing that y_{11}=1, y_{12}=1, y_{13}=0, and y_{14}=0. We know that the observed values are y_7=1 and y_1=0. Then, we will examine how correct would be to observe y_1=0, when y_7=1. Formally, we want to calculate $\text{Inf}_{v_7}(v_1)$ for this observation/execution.

It is: C_7={V_6,V_8,V_3} and D_7={V_4,V_2,V_1}.

We start by calculating all possible worlds (truth assignments) over the output values of C_7 nodes, as they resulted by output values of their parent nodes, y_{11}, y_{12}, y_{13}, and y_{14}=0. Since we are provided with an observation, we will follow the observation-based approach, as presented in section 4.1.1. Then we associate each

possible world with the respective virtual valuation. Based on these valuations, we know what the output values of D_i nodes would be.

We continue by finding the set $Vy_{C_7}^{DV}$ for the node v_7, given the observed input values of the C_7 nodes. That is, we identify out of all possible worlds over the set C_7, those where the output values of C_7 nodes, $\overrightarrow{y_{C_7}^k}$, are such, that v_7 becomes the deciding vote. These valuations would be included in $Vy_{C_7}^{DV}$.

Let $Vy_{C_7}^{DV}$={pw^1,pw^3,pw^7}, and $\overrightarrow{y_{C_7}^1}$={$y_6$=0,$y_8$=0,$y_3$=1}, $\overrightarrow{y_{C_7}^3}$={$y_6$=0,$y_8$=1,$y_3$=1}, $\overrightarrow{y_{C_7}^7}$={$y_6$=1,$y_8$=1,$y_3$=1}.

That is, for {y_6,y_8,y_3}={001,011,111}, y_7 is the deciding vote for y_1.

Also, for y_7=1, it is: $\overrightarrow{y_{D_7}^1}$={$y_4$=0,$y_2$=1,$y_1$=0}, $\overrightarrow{y_{D_7}^3}$={$y_4$=1,$y_2$=1,$y_1$=0}, $\overrightarrow{y_{D_7}^7}$={$y_4$=0,$y_2$=0,

This means that, e.g., when in pw^1, where {y_6,y_8,y_3}={001} and y_7=1, we have {y_4,y_2,y_1}={0,1,0}.

After, we sum over all possible worlds where the above hold. In this case, we would calculate:

$$\mathrm{Inf}_{y_7=1}(y_1{=}0) = \sum_{k=1,3,7} \prod_{j=1,2,3,4,6,8} \mathrm{P}(Y_j{=}y_j^k \mid \overrightarrow{y_{pa_j}^k})$$

Suppose that in our case, it is: $\mathrm{Inf}_{y_7=1}(y_1{=}0) = 0.56$.

This means that the probability that the values of C_i and D_i nodes are such, that y_7=1 would cause y_1=0 is 0.56.

Now, in order to compare this with another node's influence, we follow the exact same procedure for node v_9, when again y_{11}=1, y_{12}=1, y_{13}=0, and y_{14}=0. For this valuation let y_9=0. Suppose that we calculate that $\mathrm{Inf}_{y_9=0}(y_1{=}0) = 0.29$.

Comparing, we see that $\mathrm{Inf}_{y_7}(y_1) > \mathrm{Inf}_{y_9}(y_1)$, which means that the influence of node v_7 to node v_1 is greater than the influence of node v_9, when y_{11}=1, y_{12}=1, y_{13}=0, and y_{14}=0. That is, should we confront this observation, we may draw the conclusion that node v_7 is more probable to cause y_1=0. Informally, it is more probable that v_7 has determined y_1 and so, v_7's output should have the least possible uncertainty, in order to ensure that the "liability" of the other pipeline nodes onto node v_7 has the best possible results in terms of output quality.

Now let us use the same graph, however we will consider no uncertainty introduced by intermediate nodes/operators; all uncertainty derives from the leaf nodes. As a result, there is no point examining intermediate nodes for their influence to the root node, as their output is fully determined by the corresponding

leaf nodes(their ancestor leaf nodes). We will use our approach to compare the influence of two leaf nodes.

Suppose that we choose v_{12} and v_{13}.

It is: C_{12}=$\{v_{11},v_{13},v_{14}\}$ and C_{13}=$\{v_{11},v_{12},v_{14}\}$. As we have previously mentioned, there is no uncertainty introduced by the leaf nodes and so, all other leaf nodes of a leaf node under consideration, compose its C_i set. This is expressed by (8).

So, as in the previous case, we continue by identifying for which valuations of C_{12} nodes (possible worlds), v_{12} becomes a deciding vote, and sum over all possible worlds where this holds. This example also proves that our method reduces to the definition of influence given in [1].

4.2 Beneficial plan for node refinement

The blame we put on each influential node is the measure we use in order to decide which small subset of operators to refine in order to gain as much benefit as possible, in terms of the pipeline's output credibility.

Therefore, we suggest that if we rank the operators according to the blame we have put on them and refine the top-k selected, we will achieve to improve the pipeline's output results quality, while having the least cost: the k nodes we have suggested to refine are just a small fraction of all nodes.

4.2.1 Budget

We consider the case that we are given a budget, i.e., either a maximum number of nodes which we can refine, or an amount of training data available, which we may use in order to refine the selected nodes. The most efficient way to spend this budget is to refine the top-k blamed nodes. However, if we are not given a budget, we should calculate the trade-off between the cost of refining some nodes and the benefit we get by it and then find the optimal k. This, because after having refined some nodes (the top-ℓ) the benefit we would get by the rest might be small and thus not worth refining the rest k-ℓ nodes.

In order to do the above, we propose to quantify the benefit we get from each refinement. However, the study of the outcome of an operator refinement process is out of the scope of this thesis and as a result we leave this for future work (see Section 6.4).

4.2.2 One-at-a-time

Our approach so far examines each operator atomically. Since we measure the influence and the blame we put on each operator atomically, it is risky to apply batch refinement to the top-k nodes. The fact that refining a single node improves the result, does not imply that if we refine two or more nodes at once, this will be the most beneficial refinement plan; on the contrary, this may result in worse results than before, as performing a batch refinement may have side-effects, i.e., drastically changing the way the pipeline performs, create other/more deciding vote nodes and result in repairing a "new, non-examined" pipeline.

Refining an operator, i.e., applying further targeted training data to it in order to improve its output, may occur in the change of what the observed output values are, for some input. As a result, this would change the way a part or the whole pipeline works. As a result, in this new setting we cannot apply what we have calculated previously.

However, we may refine the top-k nodes according to our primary rank and simultaneously examine what are the consequences of each operator's refinement. Should we identify a change in a node's observed output values, we should then consider examining the case again. Else, we may continue refining the nodes following our primary rank.

We handle this problem by considering the changes that occur after refining the top-blamed node each time. After refining this node, in the new state that has emerged, we find the next node that it is more beneficial to refine. It is an iterative process in which, after every iteration we take into account the changes made by the previous iteration and continue appropriately.

We expect that the cost will not be as high as it seems, because there is high probability that the top-k blamed nodes will be in the higher levels of the graph (distance desideratum, Section 4.1.6.2). As a result, after every iteration

little changes will have occurred in the way the pipeline works. On the other hand, should the top-blamed nodes be found lower in the graph, the lower the connectivity of the graph in these lower levels, the lower would be the changes; the spread of the change will be delayed until it reaches nodes with high out-degree.

4.3 User Feedback - Debugging

In many community information management (CIM) platforms today, users not only need to evaluate their trust on the data returned, but are also frequently willing to offer to the community by flagging erroneous output. Thus, a CIM platform should provide users with the ability to give feedback about the quality of the results, report erroneous outputs, and indicate possible sources of error. Thus, the platform should exploit this valuable feedback to efficiently repair the pipeline.

Human users have expertise in a large number of heterogeneous domains. Getting feedback from users is on one hand valuable, as it indicates the correct output values and is something that usually users do not offer generously. So, when user feedback is available, such platforms should take anything they can from it. On the other hand, user feedback is sometimes biased, erroneous or may come from malicious behaviour and collusion among users.

The combination of the knowledge upon how the system works -the deciding votes- and what the correct output should be -the user feedback- can help us provide an even more beneficial improvement plan, as we would then have the posterior knowledge about what is the correct output for given input.

4.4 Guarantees

Refining a node does not indicate that its output will definitely change for the given observed input. So, we cannot assure that if we follow the proposed optimal improvement plan, we will have the desired improvement, as we cannot be sure that the refined nodes will have their CPT changed. However, we can *guarantee* that, should the output value of a node for a given input changes after the refinement, then we will have improved the system with the least possible cost.

4.5 Alternative approach for measuring influence

We provide an alternative, more naive approach for finding influential operators worth refining.

4.5.1 Toggle

This approach answers the question: "Given that we observed the input of a node, what is the best case scenario for its output to its descendant nodes, so as to lead the whole system and the root node to higher quality results?"

After an execution is performed, we are provided with the observation, i.e., the output values of all nodes. Our task is quantify the influence that the observed output value of node v_i had on the root's output y_1, i.e., how has each leaf and intermediate node's output value contributed to/affected the root's output value and probability.

In order to locate the top-k influential nodes, based on the current execution, we apply the following procedure:

We begin by introducing some sets of nodes, according to their position in the pipeline and their relative position to node v_i.

Let path(v_i,v_j) denote a path from a node v_i to a node v_j:

path(v_i,v_j)=E(v_i,v_{k_1})+E(v_{k_1},v_{k_2})+...+E(v_{k_m},v_j).

Then, the sets of Unreachable, Ancestor and Descendant nodes of a node v_i are:

- $v_j \in U_i$ if $\nexists$ path(v_i,v_j) || path(v_j,v_i) :*nodes with no connection to* v_i
- $v_j \in A_i$ if $\exists$ path(v_j,v_i) :*ancestor nodes of* v_i
- $v_j \in D_i$ if $\exists$ path(v_i,v_j) :*descendant nodes of* v_i

We denote all observed output values with $\overrightarrow{y_i}$. As every y_i has a probability to be correct, P(Y_i=y_i | $\overrightarrow{y_{pa_i}}$), we examine *what if* y_i *is wrong, i.e., what if* v_i *would output* $\neg y_i$ *for the same input* $\overrightarrow{y_{pa_i}}$*?* Therefore, we consider a "virtual" execution, where we intervene and force v_i, for the same input values as in the initial observation, $\overrightarrow{y_{pa_i}}$, to output the complement value of what we observed.

We denote $\overrightarrow{\hat{y}^i}$ the new "virtual" observation's output values, resulting from toggling y_i. The superscript here indicates which node we toggled in the current virtual observation. Then $\hat{y}_i^i$=$\neg \mathrm{y}_i$.

Node v_i can only affect -with its output- all nodes $\mathrm{v}_j \in \mathrm{D}_i$, i.e., its *space of influence* of v_i. As a result, the new observations $\overrightarrow{\hat{y}^i}$ will differ from the initially observed $\overrightarrow{y}$ at least in one value, y_i, and at most in $|\mathrm{D}_i|$ values.

We have:

- $\hat{y}_i^i = \neg \mathrm{y}_i$
- $\overrightarrow{\hat{y}^i_{U_i}} = \overrightarrow{y_{U_i}}$
- $\overrightarrow{\hat{y}^i_{A_i}} = \overrightarrow{y_{A_i}}$
- $\forall\ \mathrm{v}_j \in \mathrm{D}_i$, $\hat{y}_j^i = \mathrm{F}_{v_j}(\overrightarrow{\hat{y}^i_{pa_j}})$

The above introduce that in the new virtual observation $\overrightarrow{\hat{y}^i}$, the only nodes that have different observed output value from the initial is v_i and possibly some of the descendants of v_i, according to functions $\mathrm{F}_{v_j}(\overrightarrow{\hat{y}^i_{pa_j}})$ that describe the observed output of every $\mathrm{v}_j \in \mathrm{D}_i$, given their input $\overrightarrow{\hat{y}^i_{pa_j}}$.

We then want to calculate the probability of the new virtual observation, $\overrightarrow{\hat{y}^i}$. This is:

$$\hat{P}_i = \prod_n \mathrm{P}(\mathrm{Y}_j{=}\hat{y}_j^i \mid \overrightarrow{\hat{y}^i_{pa_j}})$$

Then we may compare each $\hat{P}_i$ with the initial probability P_0, in order to examine if toggling v_i would lead us to a more correct execution. Specifically, the difference that may occur between $\hat{P}_i$ and P_0 comes from the change that the toggle caused in D_i nodes. So, we practically examine the change of the probability of the subgraph defined by v_i and v_1, when we toggle y_i.

If $\hat{P}_i > \mathrm{P}_0$, then changing the observed output value of v_i, for its input $\overrightarrow{y_{pa_i}}$, would result in a more correct execution, for the same base input.

That is, the whole subgraph has a higher joint probability. However, making the execution more certain in total, does not indicate that we trust more the root's output, as now the root's observed input values may have changed. For this new input, the root node may be more uncertain than before.

As a result, we ask for the nodes that maximize both the probability of the nodes D_i and the root node v_1.

Based on the gain of v_i, the subgraph's and the root's output probability that we get when toggling each node's output value, we are able to rank the nodes. This ranking will provide which nodes are better candidates for refinement. However, we should clarify that refining a node does not indicate that the observed output value of the node, for the same input will change. But, we guarantee that if this output value changes, then we will have the maximum benefit in terms of root's output correctness, while having refined a small subset of the graph nodes.

Chapter 5

Experimental Evaluation

In order to prove the correctness of our definitions, we have implemented our approach in the environment of a probabilistic database management system and run a series of experiments.

We start by providing some information about the available platforms, the setting of our experiments and continue with the results.

5.1 MayBMS probabilistic database platform

MayBMS is a state-of-the-art probabilistic database management system that has been built as an extension of Postgres. It focuses on representation problems, query language design, and query evaluation

We provide here an overview:

- An extension of the Postgres server backend. Compiles and runs on the same platforms as Postgres.

- Postgres APIs and middleware can be used, e.g., ODBC, JDBC, PLSQL, PHP.

- Full SQL support. Same performance as Postgres on nonprobabilistic data.

- Full support for updates, transactions and recovery.

- Secondary storage implementations for all operations.

- Probabilistic world-set algebra: relational algebra operations + operation to compute tuple confidence + operation for introducing uncertainty

- Operations evaluated in parallel in each possible world

- U-relational databases: dbs are finite sets of possible worlds with probability weights

Much theory used for MayBMS is based on the U-Relations. Confidence computation in MayBMS can be efficiently approximated by Monte Carlo simulation.

MayBMS implements both an approximation algorithm and several exact algorithms for confidence computation. The approximation algorithm is a combination of the Karp-Luby unbiased estimator for DNF counting in a modified version adapted for confidence computation in probabilistic databases and the Dagum-Karp-Luby-Ross optimal algorithm for Monte Carlo estimation. The latter is based on sequential analysis and determines the number of invocations of the Karp-Luby estimator needed to achieve the required bound by running the estimator a small number of times to estimate its mean and variance.

The exact algorithm for confidence computation is based on an extended version of the Davis-Putnam procedure that is the basis of the best exact Satisfiability solvers in AI. Given a DNF (of which each clause is a conjunctive local condition), the algorithm employs a combination of variable elimination (as in Davis-Putnam) and decomposition of the DNF into independent subsets of clauses (i.e., subsets that do not share variables), with cost-estimation heuristics for choosing whether to use the former (and for which variable) or the latter.

An important definition in MayBMS is the repair-key operation for introducing uncertainty. The repair-by-key operation is motivated by data cleaning scenarios. When applied to a relation that violates a uniqueness constraint, a repair-by-key query generates a world-set representing all possible repairs of that relation: "An uncertainty-introducing operation, repair-key, which can be thought of as sampling a maximum repair of a key for a relation. Repairing a key of a complete relation R means to compute, as possible worlds, all subset-maximal relations obtainable from R by removing tuples such that a key constraint is satisfied."

Also, aconf(ϵ, δ) aggregate to compute an (ϵ, δ)-approximation of the probability, i.e., the probability that the computed value $\hat{p}$ returned by aconf deviates from the correct probability p by more than ϵ * p is less than δ.

MayBMS provides two extensions of SQL, "repair key" and "[a]conf". SQL extended by just these two features allows for very powerful queries, including

the computation of conditional probability tables, maximum likelihood estimates, maximum-a-posteriori, Bayesian learning, and much more.

Some more operations introduced by MayBMS are, in brief, the following:

argmax: Return the argument with the maximum value

conf: Return the exact confidence of distinct tuples

aconf: Return the approximate confidence of distinct tuples

tconf: Return the exact confidence of tuples

esum: Return the expected sum over distinct tuples

ecount: Return the expected count over distinct tuples

5.2 Experimental setting

5.2.1 Implementation choices on our approach

We have chosen to consider an approach similar to MayBMS, i.e., extending the PostgreSQL with some useful functions and operators in order to make our system more eligible and extendible. This approach would also allow us to use some of the operators introduced by MayBMS. However, unlike MayBMS, we do not use the theory of U-Relations and as a result, there is very little use of MayBMS operators in our system; we chose to implement our own model and operations in order to achieve better adaptation to our problem domain.

The reason why we have not used any of the aforementioned available systems is that the problem we study -to the best of our knowledge- has not been approached before. The nature of our problem and its special characteristics, especially the fact that we consider intermediate uncertain operators, have not allowed to use an existing system or approach.

We use PostgreSQL to describe and store the pipeline; all pipeline specifications are stored in the database. We have implemented a series of User Defined Functions (UDFs) in C and integrated them in PostgreSQL in the form of functions and operators. These UDFs allow high speed operations on the data outside the database.

A UDF is a function written outside PostgreSQL in C language and is integrated to PostgreSQL by the CREATE FUNCTION command. UDFs can take

as argument and return any base or complex type. Such functions are compiled into dynamically loadable objects (also called shared libraries) and are loaded by the server on demand. PostgreSQL also allows the user to define her own operators through the CREATE OPERATOR command. As a result, a UDF can be used to implement a user-defined operator.

We have implemented our system in C language and used the gcc compiler. The code implementation is about 4000 lines of C code, including the implementation of all approaches, graph generator and comments. We use PostgreSQL 8.3.3 to store the relations, the UDFs, and the query results. All experiments were run on a 2.1GHz Core 2 Duo machine under a VirtualBox xUbuntu 10.10, with 1Gb of dedicated main memory.

5.2.2 Data model

We use the underlying DBMS to store the pipeline. We represent each pipeline operator's topological characteristics with a table *node* {*node_id*, input_edges, output_edges, C_i, D_i} and its CPT with a table *cpt* {*cpt_id*, *node_id*(*FK*), input_values, output_value, P}. Attribute P of the *cpt* table represents the probability of the corresponding output value. The DBMS recognises the P attribute as a common floating point value. The qualitative characteristic of this value, i.e., the fact that this is a probability value, will be added by the way we handle this value afterwards. Attribute output_value of the *cpt* table refers to the observed output value. For every operator v_i, there are $2^{|pa_{v_i}|}$ tuples in the table *cpt* for each tuple of the table *node*.

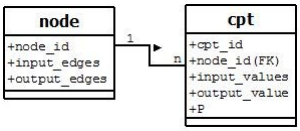


Figure 5.1: SQL schema

We use the aforementioned table representation to describe both the pipeline structure and the operators specifications. However, in order to introduce the

Bayesian network semantics, identify variable dependencies, and calculate possible worlds, conditional probabilities, calculate the influence of a node,and generally run numerous graph algorithms efficiently, we use UDFs to create a data structure that represents the directed acyclic graph. The database approach will efficiently store data and provide it, through the UDFs, to the data structure, which will in turn handle and operate on these data. Then, the results are returned to the database, allowing to run queries on them. This allows to use various graph algorithms (e.g., BFS, etc.) within a structural language (i.e., C) context, and not in the database. After having identified all useful information on graph dependencies etc., we are able to answer queries on these data by using separate UDFs.

5.2.3 Implementation details

All operators used in the pipelines for experimentation are binary and uncertain, i.e., a probability is assigned to their output values (0 or 1), for every input, to represent the false positive and false negative probabilities. The false positive and negative probabilities are uniformly distributed in [0.01,0.3].

To make things harder for our system, we have used general operators, i.e., operators that work much more complex than simple AND,OR,XOR etc. This allows to prove that our approach outperforms even in such a complex environment where nothing is obvious and complex dependencies and operators exist. Moreover, this approach is a more accurate representation of a real world system, where the operators used would be of this kind.

We have run a series of experiments assuming the following two cases: i)general setting, where no observation is available, and ii)given observations, i.e., the result of system executions over some input data. We examine both cases separately and provide the results below.

In the case where we assume that an observation is provided, the pipelines are fed with data coming from a probabilistic database, and all operators observed output values have been stored in advance. In simple words, we assume that we are provided with information from previous executions. This approach ultimately allows us to either provide the pipeline's influential operators for a

given execution, or study the pipeline in general and provide the most influential operators over all possible execution scenarios.

Also, a significant detail is that the general setting approach allows us to measure the influence of a given operator, even if we have no information about all pipeline operators *below* the given operator.

5.2.3.1 Sampling

Our approach is implemented by firstly considering the exact case, where all possible worlds are taken into account while measuring the influence of an operator.

The need to make our approach applicable to even larger pipelines has made imperative the implementation of an approximate extension to out approach that considers sampling over possible worlds. As a result, in order to calculate the influence of an operator v_i, we sample over the possible worlds of its C_i set, i.e., the possible worlds because of which v_i becomes the deciding vote.

As a result, we have much fewer possible worlds to calculate and examine and thus much less operations performed. This approximation technique makes our approach much faster, without sacrificing the quality of our results. This is better shown below, in the Experimental Results section.

In our experiments, we randomly choose the 10%, 30%, 50% or 70% of all possible worlds over the operators of the C_i set of a given operator v_i.

This approximation provides a great benefit in means of execution time, since we randomly pick some possible worlds id's and we just have to calculate these worlds and examine only these to check if the node under consideration is the deciding vote. This is better shown in the experimental results section.

5.2.3.2 User-Defined Functions

We have implemented a series of UDFs in order to:

- create the DAG structure,
- find C_i and D_i sets,
- calculate joint probabilities,

- calculate valuations and execution results,
- enumerate possible worlds,
- sample over possible worlds,
- find the deciding votes, and
- identify influential nodes.

This allows to precompute various important information, i.e., C_i and D_i sets, etc.. The returned results are stored back in the database.

The UDFs implemented are split into two main categories: UDFs to create and describe the pipeline and its characteristics as a graphical model (creatingUDFs), and UDFs to operate on the graphical model and answer queries (operatingUDFs). Whenever the pipeline is altered, so does the graphical model described by the external C language data structures and all database dependent information are updated.

CreatingUDFs populate a data structure with nodes that keep each pipeline operator's useful information: node's id, parent nodes, children nodes, nodes of C_i and D_i sets, and CPT values and probabilities.

Populating set D_i is a simple breadth-first search (BFS), starting from the target node v_i and moving up to the root. Then, the set C_i can be found by finding the direct parents of all D_i nodes and excluding from this set all D_i nodes and node v_i:

$\{C_i\}=\{pa_{D_i}\}-\{D_i\}-\{i\}$.

For the data structural approach, we use an adjacency list to denote the dependencies between nodes. This approach also guarantees low complexity of the BFS algorithm used to find D_i nodes, which is $O(|V|+|E|)$.

OperatingUDFs use data stored in the database to answer simple (e.g., find the output value of a node, given its input values) and more complex queries (e.g., find influential nodes, find deciding vote nodes, calculate possible worlds). The use of operatingUDFs is independent from the pipeline, i.e., they are able to operate on any pipeline that is stored in the database.

Briefly, we have implemented operatingUDFs to i)enumerate all possible worlds, ii)find whether an operator is the deciding vote and in which possible worlds this takes place, iii)calculate the influence of an operator, iv)generate mean output probabilities of the operators for the general setting case, v)simulate all possible execution scenarios of the pipeline, and vi)compute the influence through the heuristics approaches used for the evaluation of our results. These are the main and most important operatingUDFs.

5.2.3.3 Dataset

We have built numerous pipelines in order to experiment on, prove the correctness of our method and draw conclusions. These pipelines vary in size and characteristics. We have also implemented a pipeline generator function, which generates pipelines with the desired characteristics, for exhaustive testing.

Since there is not much work available that shares similarities to ours, there is not much information provided by the related work about the size of the graphs that other approaches use. Below, we provide some information on the datasets used by other, quite similar, approaches:

5.2.4 Graph and dataset characteristics of other approaches

- *PROBER:* In PROBER [3] is used a collection of 500 million web pages crawled by the Yahoo! search engine as dataset. They implemented two pipelines, Business and Iterative. Business pipeline contains a simple queue of 8 operators (degenerated DAG), and there is no description available for the Iterative pipeline.

- *PurpleSOX:* In PSOX prototype [4] there is no information about the size and characteristics of pipelines they plan to handle. Their paradigms show small size and low-depth pipelines, however these are used for simplifying their representation.

- In Kanagal et al. [21], they have synthesized a 100 MB TPC-H dataset augmented with tuple uncertainty for each tuple. The probabilities of existence were chosen uniformly between [0,1]. They have also built indexes

on the primary and foreign key attributes of each relation. There is also no information available on the structural characteristics of the trees used.

- *MayBMS:* In the MayBMS manual, the paradigms provided show that their testing experiments were carried out over random undirected graphs of 5-100 nodes with the same or different probabilities among the edges. Also, random directed graphs were tested, composed of 1000 to 3000 nodes, including both certain and uncertain edges, with the latter ranging from 0 to 0.1 probability.

5.3 Baseline metrics and heuristics used for evaluation

We continue by providing some simple heuristic approaches that can be used instead, in order to choose the top-k nodes for refinement. Also, we have implemented the approaches proposed by the related work. We do this in order to evaluate our technique, compare it to the heuristics results and the prior art, and show that the quality of our results outperforms simplistic approaches, providing much better cost-benefit ratio.

5.3.1 Related work-based metrics

5.3.1.1 Simple Re-Suciu [1] Influence metric

Here we calculate influence following the definition of influence provided by the paper of Re and Suciu [1], and we choose the set of top-k nodes for refinement based on each node's influence. This ignores all intermediate operators uncertainty and only chooses some leaf nodes for refinement.

5.3.2 Structure-based heuristics

5.3.2.1 Node out-degree heuristic

Using this heuristic, we rank all nodes based on the amount of outgoing edges they have (node out-degree $deg^+(v_i)$), i.e., how many child nodes they have. The

idea is that, the more nodes a node connects to, the higher its influence. This happens, because a node with high out-degree affects a larger part of the pipeline, i.e., many other nodes.

5.3.2.2 Node distance heuristic

This heuristic is based on the distance of a node from the root node. We consider that, the shorter the distance to the root, the higher the influence. If a node is very close to the root, its decision (output) cannot be overshadowed by many other nodes and thus, it is more possible to influence the root node.

Moreover, since the pipelines we consider result in a single root, this means that while moving up in the pipeline, the pipeline narrows. As a result, there might be only few other nodes connecting to the root in short distance. This indicates that we should expect highly influential nodes closely to the root node.

5.3.2.3 Combined out-degree and distance heuristic

This heuristic combines the above two. It considers nodes that are both in short distance to the root node and have high out-degree. The following formula is used to rank the nodes:

$\text{Influence}_{combo\_heuristic}(\text{v}_i) = 2^{distance(max)-distance(v_i)}+deg^+(v_i)$

where distance(max) is the distance of the node with the maximum distance from the root node in the pipeline under consideration.

The above heuristic emphasizes the role of the node's distance, as we expect that the node distance is more important than the out-degree.

5.3.3 Summary

To sum up all the parameters considered for experimentation, we have:

We consider the following two cases:

i) General setting, where neither observed output data nor input data are available. As a result we run a series of experiments while having computed a mean probability for the input values of a node under consideration, based on the false positive/negative probabilities of its parents. In this case, we reason about

the influence of the pipeline operators in the general case, not based on one or some observations.

ii) Given observation, where we are provided with either some input data, or data coming from previous executions. In this case, we compute each node's influence based on the data provided. For each pipeline, we show the aggregated results for all input datasets used.

Also, for each of the above cases, we have tested our system for both the exact and the approximate case:

i) The exact case enumerates all possible worlds for which a node is the deciding vote.

ii) The approximate case samples 30%, 50% and 70% of the above worlds.

We continue by providing some plots to show the scalability of the model and how the influence of a node is affected by various parameters. The following results confirm our intuitions, prove that our approach outperforms both the heuristic and the simplistic approaches, and show that our approach and implementation can also be used in the operator influence analysis of large pipelines.

5.4 Experimental results

After running a series of experiments, the setting of which is described in section 5.2, we provide and comment the results.

We provide the experimental results for each pipeline tested, separately.

Root node is denoted as v_0, instead of v_1 that was used so far, in every pipeline.

We remind that by definition (Section 4.1.9), the influence of the root node v_0 is: $\text{Inf}(v_0) = 1$.

Time is measured in seconds, in all figures.

The execution times provided, do not contain the time needed to transfer the data from PostgreSQL and create the graphical model, i.e., the execution time of createUDFs. This stage was separately executed, once for each pipeline, and the time needed was from 0.001 to 0.02 seconds for the following pipelines, depending on the size and the pipeline operators topological characteristics.

We consider the ground truth, with which we will compare the quality of results of each method, the case where we are given all possible valuations of

all pipeline operators, i.e., all possible execution scenarios. On these data, we perform exact calculation of the influence of each operator, given observations. Since we use the "exact, observation given" approach for the ground truth, the results by this method are not included in the following diagrams.

5.4.1 Pipeline no.1

Figure 5.2 provides a visualization of the pipeline:

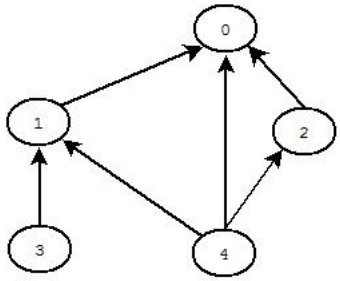


Figure 5.2: Pipeline no.1

5.4.1.1 Pipeline characteristics and structure

Pipeline no.1 (Fig.5.2) consists of 5 nodes, 6 edges, has 2 leaf nodes and the distance(max) is 2.

We can identify that the leaf nodes are the nodes 3 and 4.

Figure 5.3 shows the C_i and D_i sets for each node of the pipeline:

node_id	Ci	Di
0	-	-
1	2,4	0
2	1,4	0
3	2,4	0,1
4	3	0,1,2

Figure 5.3: Ci and Di sets of pipeline no.1

Figure 5.4 provides the conditional probability tables of all pipeline operators. The input_value column denotes the decimal representation of the input values

combination, i.e., input_value=3 for node 0 denotes that the input values in this case is "011", or y_1=0, y_2=1, and y_4=1.

CPTs

Node	Input_nodes	Input_values	Output_values	Output_prob
0	1,2,4	0	1	0.8
		1	0	0.85
		2	1	0.93
		3	1	0.7
		4	0	0.75
		5	0	0.85
		6	1	0.75
		7	0	0.95
1	3,4	0	1	0.98
		1	0	0.85
		2	1	0.75
		3	1	0.7
2	4	0	0	0.7
		1	1	0.8
3	-	0	0	0.85
		1	1	0.99
4	-	0	0	0.7
		1	1	0.98

Figure 5.4: CPTs of pipeline no.1

We continue by providing figure 5.5, which shows the number of all possible worlds over this pipeline, and also the number of possible worlds for each node separately. We should remind that $\#\text{pw}_{v_i}=2^{|C_i|}$.

Graph stats	
#PW(graph)	32
#PW(C1)	4
#PW(C2)	4
#PW(C3)	4
#PW(C4)	2

Figure 5.5: Possible worlds over all C_i sets for pipeline no.1

5.4.1.2 Execution time

Figure 5.6 provides the execution time needed to perform the following actions, i.e., run the corresponding operateUDFs:

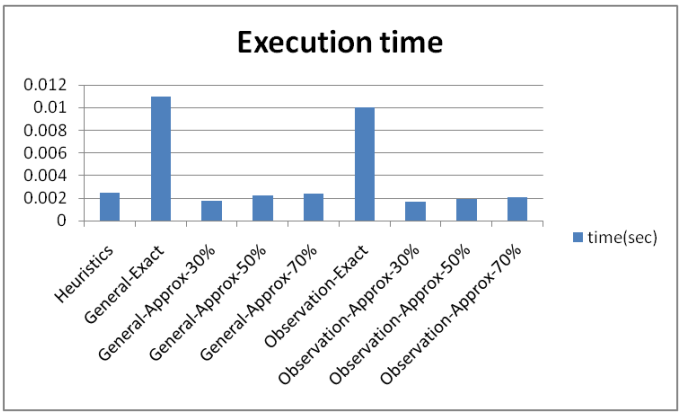


Figure 5.6: Execution time for pipeline no.1

The execution time needed for each operation agrees with our prior expectation; we expected that the time needed to compute the heuristics would be much smaller than the time needed for the exact approaches, and quite similar to the time needed for the approximate approach. The execution time for the general case is slightly higher than the time needed by the observation-based approach, since we have to calculate the mean output probability for every node, for each output value.

5.4.1.3 Ground truth

The results considered as the ground truth for this pipeline are shown in figure 5.7:

GROUND TRUTH
rank
obs-exact
0
1
3
2
4

Figure 5.7: Ground truth for pipeline no.1

As the ground truth for the influence rank, we consider the exact approach over all possible observations.

5.4.1.4 Heuristics

Figure 5.8 provides the ranking of the nodes based on their influence, as it has been computed by using each of the heuristics:

Pipeline no.1			
Influence rank	Heuristic-combo	Heuristic-out-degree	Heuristic-Distance
1	4	4	0
2	0	3,1,2	4,1,2
3	2,1	0	3
4	3		

Figure 5.8: Results of heuristics for pipeline no.1

5.4.1.5 Our approach

Continuing with our approach, figure 5.9 shows all nodes and their influence values, sorted from higher to lower influence:

EXACT rank	General Influence	Observation Influence	EXACT rank
0	1	1	0
1	0.164795	0.2622985	1
2	0.130703	0.1646695	3
4	0.10532	0.1473485	2
3	0.089394	0.1269525	4

APPROX-50 rank	General Influence	Observation Influence	APPROX-50 rank
0	1	1	0
1	0.090964	0.13024525	1
2	0.05921	0.08023925	3
4	0.0495	0.075955	2
3	0.048801	0.0640765	4

APPROX-30 rank	General Influence	Observation Influence	APPROX-30 rank
0	1	1	0
1	0.038432	0.275917	1
2	0.033109	0.179152	3
3	0.019667	0.17412	2
4	0	0	4

APPROX-70 rank	General Influence	Observation Influence	APPROX-70 rank
0	1	1	0
1	0.081396	0.13835825	1
2	0.065788	0.0849745	3
4	0.05582	0.071654	2
3	0.041978	0.06412825	4

Figure 5.9: Influence values of all nodes, sorted by influence, for pipeline no.1

Figure 5.10 provides the ranking of the nodes, as resulted by the computation of influence by following both the exact and the approximate approaches, and for all sampling ratios used.

	EXACT	APPROX-30		APPROX-50		APPROX-70	
Influence rank	General	General	Observation	General	Observation	General	Observation
1	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1
3	2	2	3	2	3	2	3
4	4	3	2	4	2	4	2
5	3	4	4	3	4	3	4

Figure 5.10: Node ranking by influence for exact and approximate cases, for both general and given observation cases, for pipeline no.1

Figure 5.11 provides information about how much each approach verges on the ground truth provided above.

The first column (pos) includes in which positions of the ranking each approach's result matches with the ground truth. As a result, the second column (hits) shows the amount of nodes found in the same ranking position with the ground truth, i.e., in which nodes' ranking do the ground truth and each approach agree.

The last three columns show how much each approach achieved to agree with the ground truth at the top 25% nodes of the pipeline, the top 40% and finally at the top influential node.

	Pipeline no.1				
	Position of hits	Hits(%nodes,position)	top-25%	top-40%	top-1
Heuristic-combo	-	0	0	0	0
Heuristic-Out degree	4	0.2	0	0	0
Heuristic-Distance	4	0.2	0	0	0
General-Exact	0,1	0.4	1	1	1
General-Approx-30%	0,1	0.4	1	0.8	1
Observation-Approx-30%	0,1,2,3,4,5	1	1	1	1
General-Approx-50%	0,1	0.4	1	1	1
Observation-Approx-50%	0,1,2,3,4,5	1	1	1	1
General-Approx-70%	0,1	0.4	1	1	1
Observation-Approx-70%	0,1,2,3,4,5	1	1	1	1

Figure 5.11: Figure of the quality of results for all cases, for pipeline no.1

As we may observe, our approach outperforms in every case the heuristic approaches. None of the heuristics approaches managed to identify even the top influential node, which is the root node.

Among our results, we can identify that observation-given cases achieve better results than the general case. This happens due to the fact that in the general case we calculate the mean probability for each node's output value, thus we just have a general notion on the operators false positive/negative ratios.

If we take a closer look at the node rankings by the general case, even with approximations, we can see that although we have a low hit rate (40%), each node's distance from its correct position in the ranking is not more than 1 position, in most of the cases. Moreover, this malfunction takes place mostly below the top-40% influential nodes ranking, meaning that in almost the top half of the ranking, we have totally correct results.

The above conclusions can be better confirmed by the following diagrams in figures 5.12, 5.13, 5.14, and 5.15:

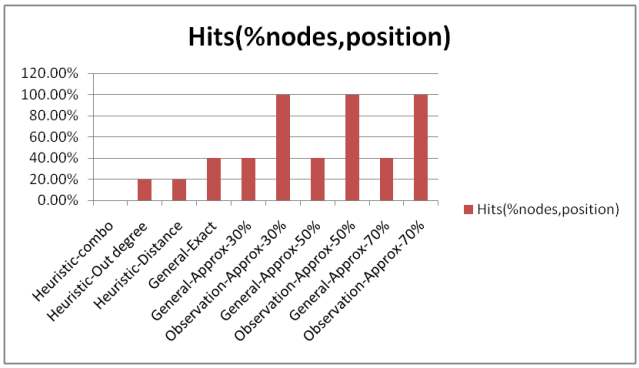


Figure 5.12: Hit rate for pipeline no.1

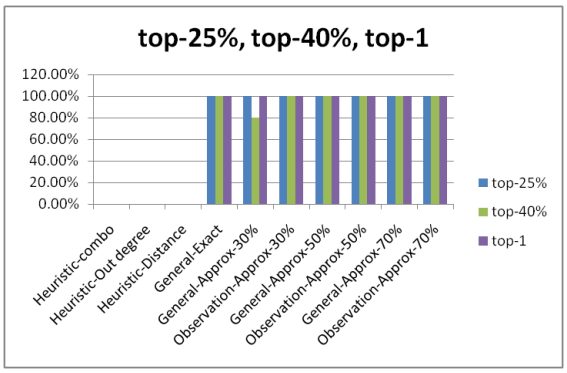


Figure 5.13: Top-25%, top-40% and top-1 results for pipeline no.1

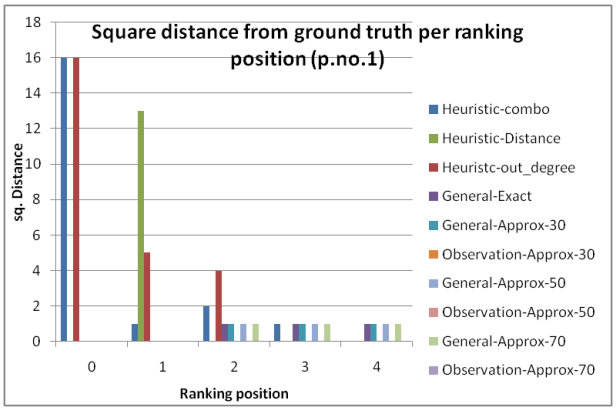


Figure 5.14: Square distance from ground truth per ranking position

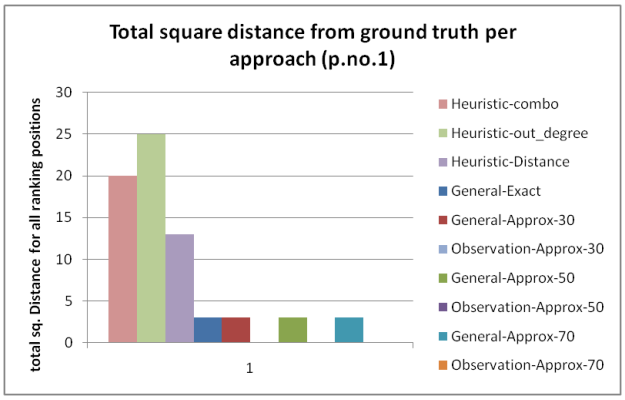


Figure 5.15: Total square distance from ground truth per approach

The following two figures (5.16 and 5.17) show each node's influence value convergence to the influence value by the ground truth, as calculated by each approximation approach.

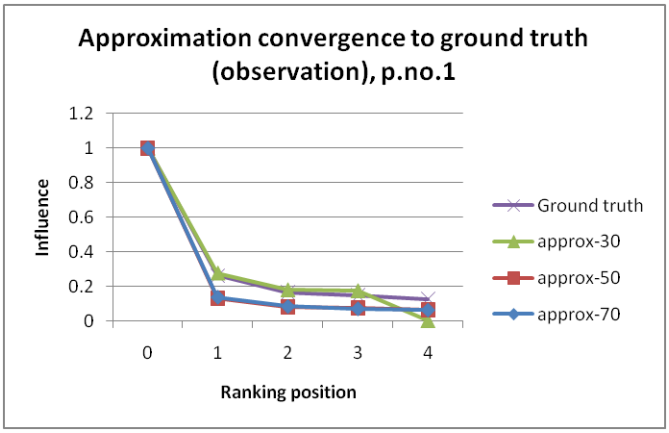


Figure 5.16: Influence value convergence to ground truth value for each approximation approach, observation-based approach

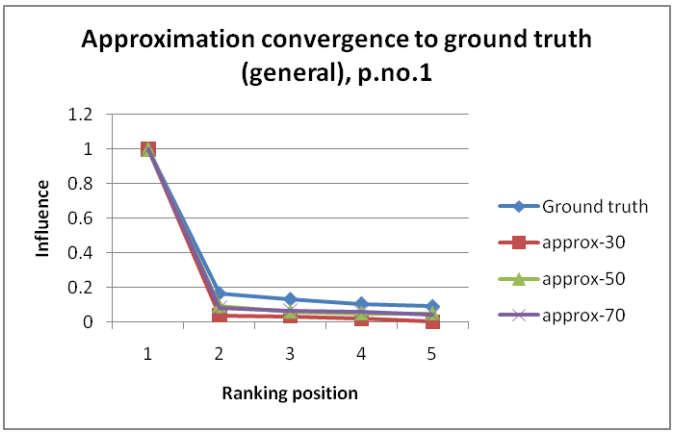


Figure 5.17: Influence value convergence to ground truth value for each approximation approach, general case

The results of figure 5.16 might be quite misleading though; we assume that approx-30 approach was enough "lucky" to have sampled all possible worlds for which each operator is the deciding vote and, as a result, went so well calculating the influence value for each node, apart from node 4. Approx-30 failed to calculate node's 4 influence value due to the very few possible worlds of node 4.

The following figures (5.18 and 5.19) show how the influence value is correlated to each node's C_i and D_i sets size.

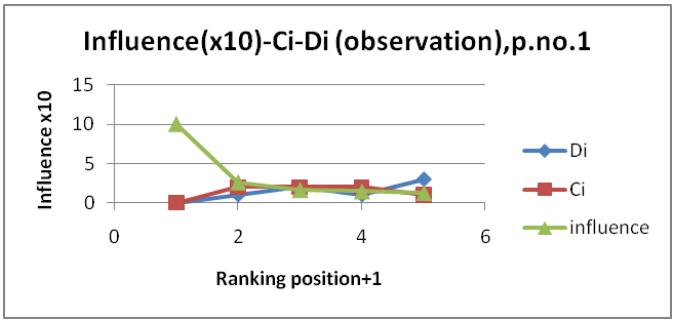


Figure 5.18: Influence value correlation to size of sets C_i and D_i, observation-based approach

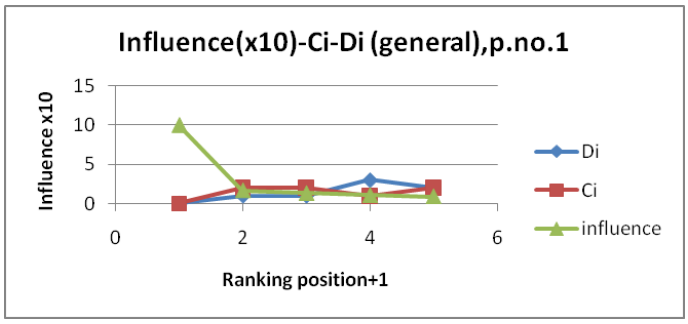


Figure 5.19: Influence value correlation to size of sets C_i and D_i, general case

Due to the fact that pipeline no.1 is quite small, it is not yet clear how the size of the sets C_i and D_i affects the influence value; however, this is better shown in the following pipelines.

5.4.1.6 Concluding remarks

The conclusions drawn by pipeline no.1 show that our approach outperforms in means of quality of the results, in comparison to the heuristic approaches. The execution time of our approach is higher, however we manage to have almost top-quality results, when most heuristic approaches almost completely fail.

This first pipeline presented, also gives us the opportunity to discuss an important aspect of the influence computation. Studying the above pipeline in detail, we should recognise that not the pipeline structure, but mostly each operator's conditional probability table, determines if a given operator would have high or low influence. In simple words, it is not how the operators interconnect, but their IO-specifications (see section 3.3.2) and their false positive/negative ratios that determine if and how much is each node influential.

This can be better understood by taking a closer look at pipeline no.1: most users would argue that the most influential node would be node 4. This node seems more "central" or important in the pipeline. However, looking at how its neighbouring operators work, we will see that the other nodes practically "ignore" its output, or to put it right, there are very few cases, very few possible worlds in which node 4 is the deciding vote. In short, should we replace one or more operators, i.e., change their CPT and their false positive/negative ratios but keep

the same pipeline structure, the heuristic approaches would still output the above results, where the most influential nodes of the pipeline would have probably been altered. This change is captured by our approaches, where an operator replacement would immediately yield revision of the influence of all operators.

5.4.2 Pipeline no.2

Figure 5.20 shows pipeline no.2. This pipeline consists of 8 nodes, 10 edges, has 2 leaf nodes and the distance(max) is 4.

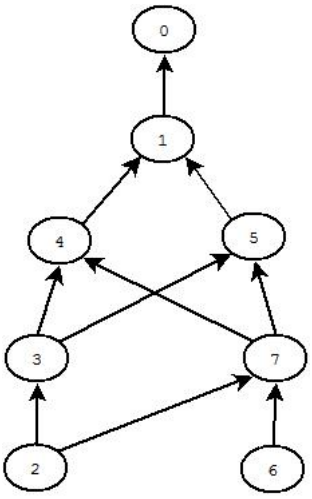


Figure 5.20: Pipeline no.2

5.4.2.1 Pipeline characteristics and structure

Figure 5.21 shows the C_i and D_i sets for each node of the pipeline:

node_id	Ci	Di
0	-	-
1	-	0
2	6	3,7,4,5,1,0
3	7	4,5,1,0
4	5	1,0
5	4	1,0
6	2,3	7,4,5,1,0
7	3	4,5,1,0

Figure 5.21: Ci and Di sets of pipeline no.2

The next figure, 5.22, provides the conditional probability tables of all pipeline operators. The input_value column denotes the decimal representation of the input values combination, i.e., input_value=2 for the node 1 denotes that the input values in this situation is "10", or y_4=1 and y_5=0.

CPTs

Node	Input_nodes	Input_values	Output_value	Output_prob
0	1	0	0	0.749497
		1	1	0.955855
1	4,5	0	1	0.838275
		1	0	0.856738
		2	1	0.902362
		3	1	0.922221
2	-	0	1	0.972229
		1	0	0.779915
3	2	0	0	0.843398
		1	0	0.898751
4	3,7	0	0	0.903847
		1	0	0.708547
		2	1	0.902362
		3	1	0.922221
5	3,7	0	1	0.90039
		1	0	0.844065
		2	1	0.976328
		3	0	0.950371
6	-	0	0	0.797485
		1	1	0.803091
7	2,6	0	1	0.882569
		1	1	0.972471
		2	1	0.757747
		3	0	0.869488

Figure 5.22: CPTs of pipeline no.2

We continue by providing figure 5.23, which shows the number of all possible

worlds over this pipeline, and also the number of possible worlds for each node separately. We should remind that $\#\text{pw}_{v_i}=2^{|C_i|}$.

Graph stats	
#PW(graph)	256
#PW(C1)	0
#PW(C2)	4
#PW(C3)	4
#PW(C4)	4
#PW(C5)	4
#PW(C6)	2
#PW(C7)	4

Figure 5.23: Possible worlds for pipeline no.2

5.4.2.2 Execution time

We continue with figure 5.24, which provides the execution time needed to perform the following actions, i.e., run the corresponding operateUDFs:

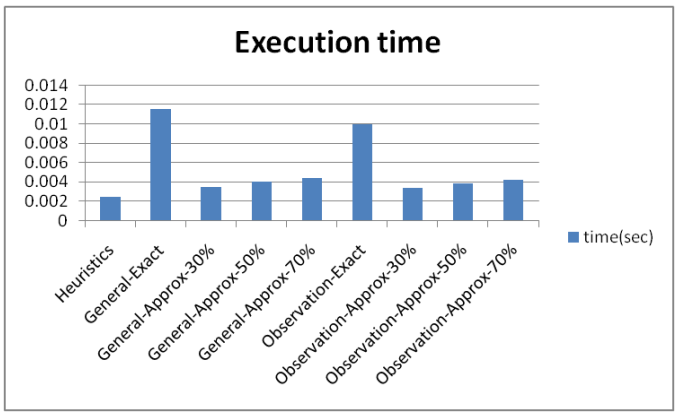


Figure 5.24: Execution time for pipeline no.2

Again in this pipeline, the execution time needed for each operation agrees with our prior expectation, that the time needed to compute the heuristics would be much smaller than the time needed for the exact approaches, and quite similar to the time needed for the approximate approach. The execution time for the general case is also slightly higher than the time needed by the observation-based

approach, since we have to calculate the mean output probability for every node, for each output value.

5.4.2.3 Ground truth

The results considered as the ground truth for this pipeline are shown in figure 5.25:

GROUND TRUTH
rank
obs-ex
0
1
5
4
7
2
6
3

Figure 5.25: Ground truth for pipeline no.2

As the ground truth for the influence rank, we consider the exact approach over all possible observations.

5.4.2.4 Heuristics

Figure 5.26 provides the ranking of the nodes based on their influence, as it has been computed by using each of the heuristics:

Pipeline no.2			
Influence rank	Heuristic-combo	Heuristic-out-degree	Heuristic-Distance
1	0	2,7,3	0
2	1	4,6,5,1	1
3	5,4	0	4,5
4	3,7		7,3
5	2		2,6
6	6		

Figure 5.26: Results of heuristics for pipeline no.2

5.4.2.5 Our approach

Continuing with the results of our approach, figure 5.27 shows all nodes and their influence values, sorted from higher to lower influence:

EXACT rank	General Influence	Observation Influence	EXACT rank
0	1	1	0
1	0.8	0.79780925	1
7	0.214646	0.50594725	5
4	0.201595	0.35085925	4
5	0.16055	0.261602	7
3	0.087234	0.100839	2
2	0.080556	0.086973	6
6	0.070354	0.02428175	3

APPROX-30 rank	General Influence	Observation Influence	APPROX-30 rank
0	1	1	0
1	0.32	0.3253	1
7	0.096591	0.1896	7
4	0.110877	0.1343	4
5	0.072247	0.1177	5
6	0.039255	0.0394	6
3	0.036165	0.03144925	2
2	0.034104	0.0182	3

APPROX-50 rank	General Influence	Observation Influence	APPROX-50 rank
0	1	1	0
1	0.38	0.3725	1
7	0.100884	0.24965625	5
4	0.106845	0.1828225	4
5	0.075459	0.12839925	7
6	0.048801	0.059451	6
2	0.042695	0.0500135	2
3	0.041	0.012659	3

APPROX-70 rank	General Influence	Observation Influence	APPROX-70 rank
0	1	1	0
1	0.38	0.3725	1
7	0.118055	0.24633825	5
4	0.090718	0.17656225	4
5	0.088303	0.12388225	7
3	0.047979	0.01147375	3
6	0.044391	0.05552375	6
2	0.03625	0.05243825	2

Figure 5.27: Influence values of all nodes, sorted by influence, for pipeline no.2

Figure 5.28 provides the ranking of the nodes, as resulted by the computation of influence by following both the exact and the approximate approaches, and for all sampling ratios used.

	EXACT	APPROX-30		APPROX-50		APPROX-70	
Influence rank	General	General	Observation	General	Observation	General	Observation
1	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1
3	7	5	7	7	5	7	5
4	4	4	4	4	4	4	4
5	5	7	5	5	7	5	7
6	3	6	6	6	6	3	3
7	2	3	2	2	2	6	6
8	6	2	3	3	3	2	2

Figure 5.28: Node ranking by influence for exact and approximate cases, for both general and given observation cases, for pipeline no.2

Figure 5.29 provides information about how much each approach verges on the ground truth provided above.

The first column (pos) includes in which positions of the ranking each approach's result match with the ground truth. As a result, the second column (hits) shows the amount of nodes found in the same ranking position with the ground truth, i.e., in which nodes' ranking do the ground truth and each approach agree.

The three last columns show how much each approach achieved to agree with the ground truth at the top 25% nodes of the pipeline, the top 40% and finally at the top influential node.

	Pipeline no.2				
	Position of hits	Hits(%nodes,position)	top-25%	top-40%	top-1
Heuristic-combo	0,1,	0.25	1	0.5	1
Heuristic-Out degree	-	0	0	0	0
Heuristic-Distance	0,1	0.25	1	0.5	1
General-Exact	0,1,3	0.375	1	0.75	1
General-Approx-30%	0,1,3	0.375	1	0.75	1
Observation-Approx-30%	0,1,3,7	0.5	1	0.75	1
General-Approx-50%	0,1,3,7	0.5	1	0.75	1
Observation-Approx-50%	0,1,2,3,4,7	0.75	1	1	1
General-Approx-70%	0,1,3,6	0.5	1	0.75	1
Observation-Approx-70%	0,1,2,3,4,6	0.75	1	1	1

Figure 5.29: Figure of the quality of results for all cases, for pipeline no.2

Figure 5.29 shows that our approach outperforms the heuristic approaches, in all cases.

The heuristic approaches have mostly failed in all aspects. The worst results were returned by out_degree_heuristic, which achieved no hits and even failed to identify the top influential node, i.e., the root node. Distance_heuristic only achieved 25% hits, however we should mention that it did well finding top-25% of the influential nodes. Finally, only combo_heuristic managed to have 50% hits.

Among our results, we can identify that again, observation-given cases achieve better results than the general case. This happens, due to the fact that in the general case we have to calculate the mean probability for each node's output value. However, even in the general case with 50% sampling which achieved the worst results among our approaches, these results outperform combo_heuristic in terms of top-40% influential nodes.

The above conclusions can be better understood by the diagrams in figures 5.30, 5.31, 5.32, and 5.33:

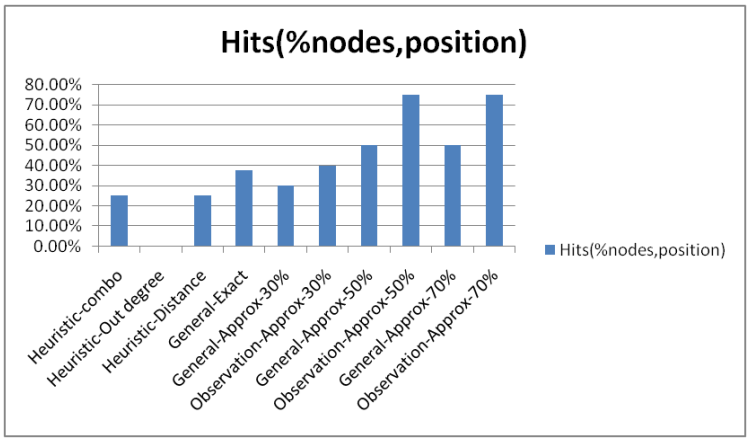


Figure 5.30: Hit rate for pipeline no.2

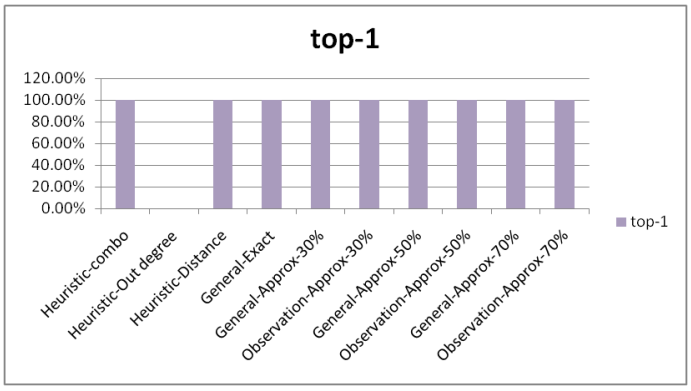


Figure 5.31: Top-1 influential node results for pipeline no.2

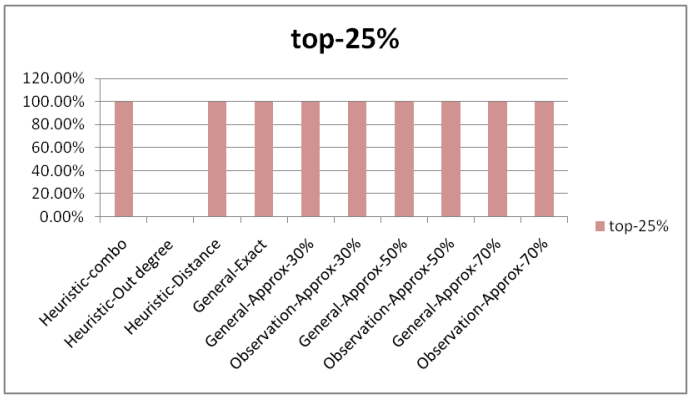


Figure 5.32: Top-25% influential node results for pipeline no.2

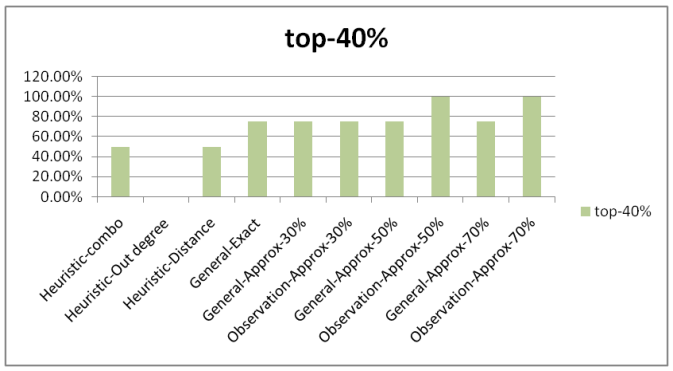


Figure 5.33: Top-40% influential node results for pipeline no.2

We continue by providing some diagrams which show the square distance from the ground truth, per ranking position, for each approach (Fig. 5.34) and the total square distance of all rankings for each approach (Fig. 5.35).

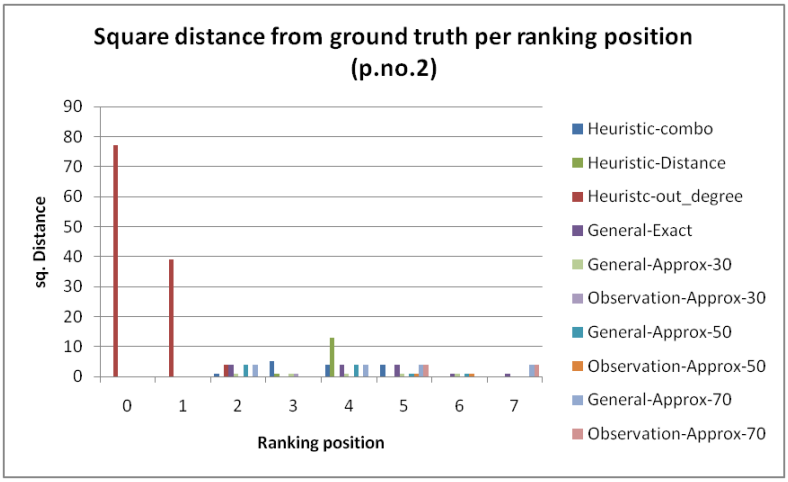


Figure 5.34: Square distance from ground truth per ranking position

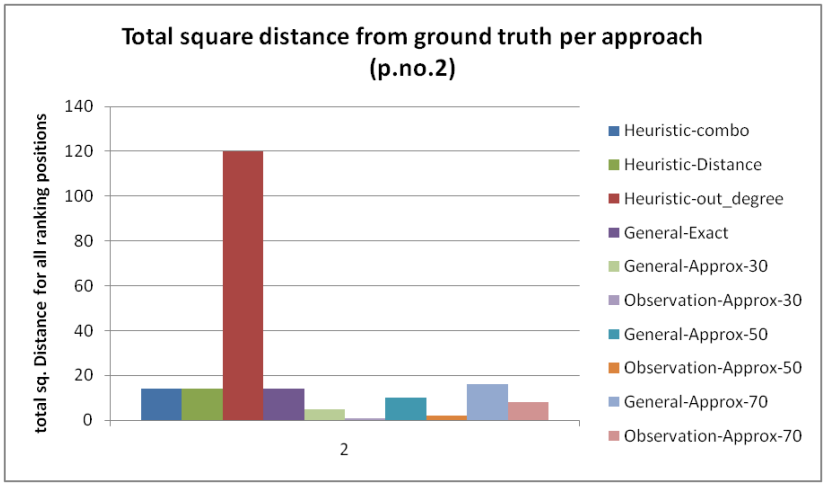


Figure 5.35: Total square distance from ground truth per approach

Figure 5.35 shows that two out of three heuristic approaches did quite well in terms of total square distance of the returned ranking positions from the ground truth. This means that these two techniques ranked almost correctly the nodes

by their influence. Also, figure 5.32 shows that the heuristic approaches also performed well in ranking highly influential nodes.

However, figures 5.30 and 5.33 clearly show that the heuristics only perform well in only the few first ($\leq$25%) ranking positions. Going down the ranking, we observe that the heuristic approaches do not perform well at all. As a result, heuristics fail to rank correctly most of the pipeline nodes, whereas our approach succeeds in all ranking positions, in all cases, by distributing a very small ranking error into all ranking positions.

The following two figures (5.36 and 5.37) show each node's influence value convergence to the influence value by the ground truth, as calculated by each approximation approach.

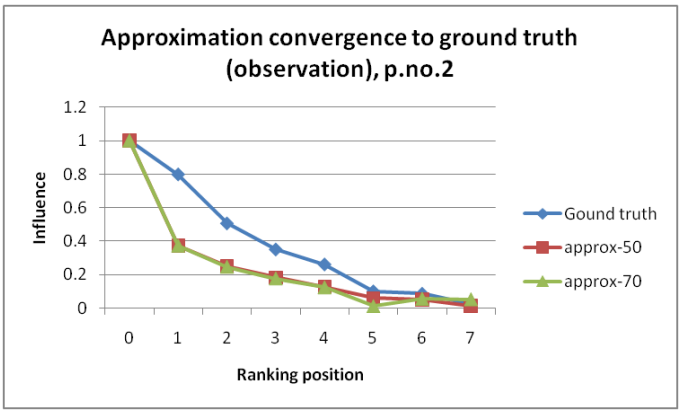


Figure 5.36: Influence value convergence to ground truth value for each approximation approach, observation-based approach

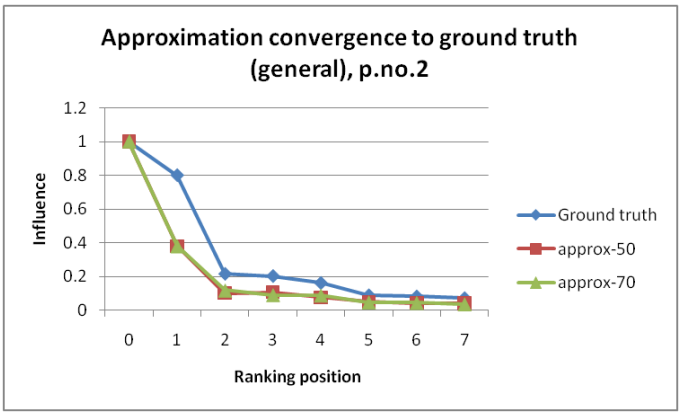


Figure 5.37: Influence value convergence to ground truth value for each approximation approach, general case

We may observe that both approximation approaches had provided similarly high quality results.

The following figures (5.38 and 5.39) show how the influence value is correlated to each node's C_i and D_i sets size.

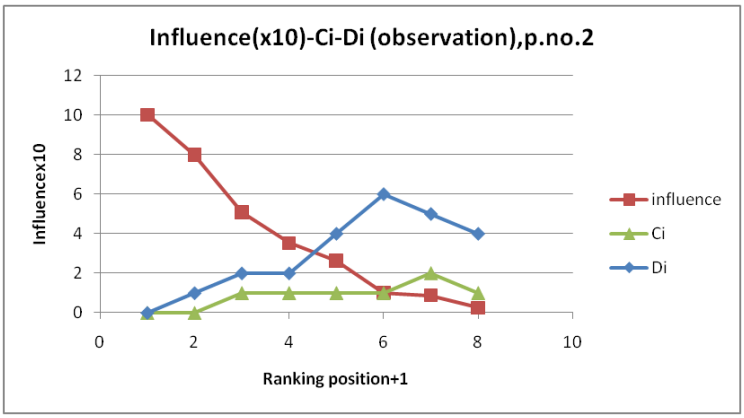


Figure 5.38: Influence value correlation to size of sets C_i and D_i, observation-based approach

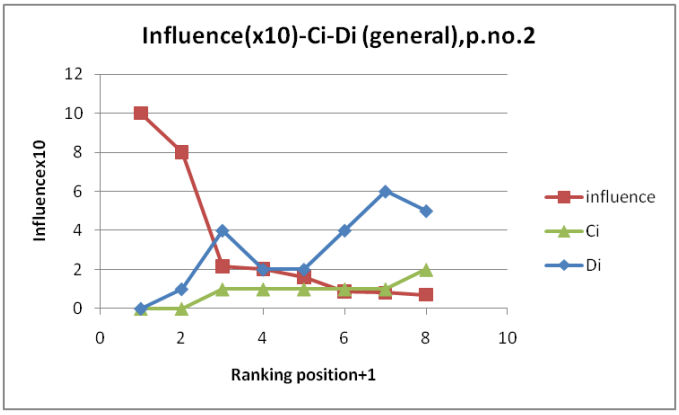


Figure 5.39: Influence value correlation to size of sets C_i and D_i, general case

The above diagrams prove that the influence value is closely related to the size of the set D_i, at least in such small pipelines where C_i set's size is quite small. This is expected, as the larger the D_i set, the more nodes we have to assign True (see Section 4.1.3.3).

5.4.2.6 Concluding remarks

One useful conclusion that we can draw by this pipeline, is that small-sized pipelines, and especially the narrow ones (small C_i set's size), need larger sampling ratios ($\geq$50%) in order to return satisfying results. This happens because such pipeline nodes have only a few nodes in their C_i sets and thus, lower sampling rates are not able to find possible worlds in which a node under consideration is the deciding vote. On the other hand, as we will show in the next pipeline, larger graphs allow to have good results even with low sampling ratios (eg. 30%), since it is more probable to sample a possible world were a node under consideration is the deciding vote, should there be many such possible worlds.

In terms of the quality results by our approaches, pipeline no.2 also confirms that our method outperforms all heuristics, since our worst returned results (approximation with 50% of possible worlds sampled in the general case) match with the best returned results of the heuristic approaches (combo_heuristic).

Again, the observation-given approach managed to do better than the general case approach; the cause of this has been thoroughly discussed earlier.

5.4.3 Pipeline no.3

Pipeline no.3, shown in figure 5.40, consists of 8 nodes, 11 edges, has 5 leaf nodes and the distance(max) is 2.

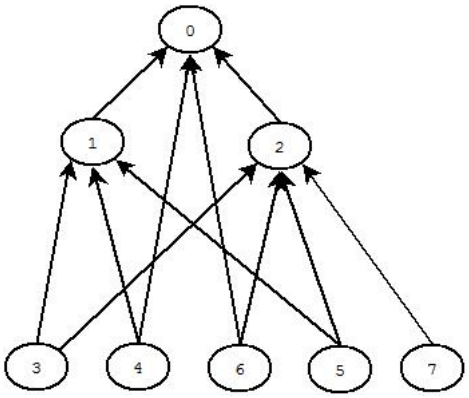


Figure 5.40: Pipeline no.3

5.4.3.1 Pipeline characteristics and structure

Figure 5.41 shows the C_i and D_i sets for each node of the pipeline:

node_id	Ci	Di
0	-	-
1	2,4,6	0
2	1,4,6	0
3	4,5,6,7	0,1,2
4	2,3,5,6,7	0,1
5	3,4,6,7	0,1,2
6	1,3,4,5,7	0,2
7	1,3,4,5,6	0,2

Figure 5.41: Ci and Di sets of pipeline no.3

The next figure, 5.42, provides the conditional probability tables of all pipeline operators. The input_value column denotes the decimal representation of the input values combination, i.e., input_value=9 for the node 0 denotes that the input values in this situation is "1001", or y_1=1, y_2=0, y_4=0, and y_6=1.

CPTs

Node	Input_nodes	Input_values	Output_value	Output_prob
0	1,2,4,6	0	1	0.749497
		1	1	0.955855
		2	1	0.838275
		3	0	0.856738
		4	1	0.902362
		5	1	0.922221
		6	0	0.843398
		7	0	0.898751
		8	0	0.903847
		9	0	0.708547
		10	1	0.902362
		11	1	0.922221
		12	1	0.90039
		13	0	0.844065
		14	1	0.976328
		15	1	0.972229
1	3,4,5	0	1	0.838275
		1	0	0.856738
		2	1	0.902362
		3	1	0.922221
		4	1	0.757747
		5	0	0.869488
		6	1	0.757747
		7	0	0.869488
2	3,5,6,7	0	1	0.749497
		1	1	0.955855
		2	1	0.838275
		3	0	0.856738
		4	1	0.902362
		5	1	0.922221
		6	0	0.843398
		7	0	0.898751
		8	0	0.903847
		9	0	0.708547
		10	1	0.902362
		11	1	0.922221
		12	1	0.90039
		13	0	0.844065
		14	1	0.976328
		15	1	0.972229
3	-	0	0	0.843398
		1	0	0.898751
4	-	0	0	0.903847
		1	0	0.708547
5	-	0	1	0.976328
		1	0	0.950371
6	-	0	0	0.797485
		1	1	0.803091
7	-	0	1	0.882569
		1	1	0.972471

Figure 5.42: CPTs of pipeline no.3

We continue by providing Figure 5.43, which shows the number of all possible worlds over this pipeline, and also the number of possible worlds for each node separately. We should remind that $\#\text{pw}_{v_i}=2^{|C_i|}$.

Graph stats	
#PW(graph)	256
#PW(C1)	8
#PW(C2)	8
#PW(C3)	16
#PW(C4)	32
#PW(C5)	16
#PW(C6)	32
#PW(C7)	32

Figure 5.43: Possible worlds for pipeline no.3

5.4.3.2 Execution time

We continue with figure 5.44, which provides the execution time needed to perform the following actions, i.e., run the corresponding operateUDFs:

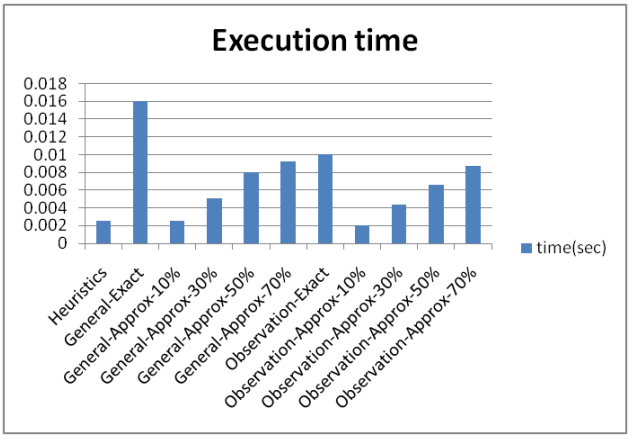


Figure 5.44: Execution time for pipeline no.3

Pipeline no.3 differentiates from the previous two pipelines, in terms of the size of most C_i node sets. In this pipeline, the size of C_i sets varies from 3 to 5 nodes, i.e., 8 to 32 possible worlds need to be calculated and examined for each node in order to discover if and in which possible worlds the deciding vote criterion

holds. As a result, the execution time for the exact case, for both the general and the observation-based scenarios, are high. Even the low cost approximation approach with 30% of possible worlds sampling ratio costs about twice as much as the heuristics.

Because of the aforementioned increased size of most C_i sets in pipeline no.3, we have chosen to further reduce the possible worlds sampling ratio for this pipeline. As a result, we have also used 10% sampling ratio to approximate possible worlds.

Figure 5.44 shows that in the approximate-10% case, the execution time is less or equal to the execution time needed by the heuristic approaches.

5.4.3.3 Ground truth

The results considered as the ground truth for this pipeline are shown in figure 5.45:

GROUND TRUTH
rank
obs-ex
0
1
6
5
2
4
7
3

Figure 5.45: Ground truth for pipeline no.3

As the ground truth for the influence rank, we consider the exact approach over all possible observations.

5.4.3.4 Heuristics

Figure 5.46 provides the ranking of the nodes based on their influence, as it has been computed by using each of the heuristics:

Pipeline no.3			
Influence rank	Heuristic-combo	Heuristic-out-degree	Heuristic-Distance
1	6,0,4	4,6,5,3	0
2	3,1,5,2	2,1,7	4,6,2,1
3	7	0	5,7,3

Figure 5.46: Results of heuristics for pipeline no.3

5.4.3.5 Our approach

Continuing with the results of our approach, figure 5.47 shows all nodes and their influence values, sorted from higher to lower influence:

rank	General Influence	EXACT	Observation Influence	rank
0	1		1	0
1	0.308778		0.18846016	1
5	0.147068		0.119535	6
4	0.140612		0.057164	5
6	0.128514		0.03327722	2
2	0.120838		0.02913675	4
3	0.087047		0.01466825	7
7	0.005901		0.011497	3

rank	General Influence	APPROX-10	Observation Influence	rank
0	1		1	0
1	0.0429117		0.02986547	2
6	0.0017318		0.03025441	1
4	0.0104752		0.00076619	6
3	0.0143805		0.00118969	5
5	0.0033897		0.00628572	4
2	0.019086		0.02032353	3
7	0.0001515		0	7

rank	General Influence	APPROX-30	Observation Influence	rank
0	1		1	0
1	0.081627		0.05142606	1
5	0.030659		0.02493081	6
6	0.024772		0.00283425	2
4	0.033858		0.00742547	5
2	0.038797		0.01563294	4
3	0.038409		0.03454791	7
7	0.001343		0.00400272	3

rank	General Influence	APPROX-50	Observation Influence	rank
0	1		1	0
1	0.159785		0.11253741	1
5	0.085541		0.06262516	6
6	0.071001		0.05092816	2
4	0.067434		0.03197853	5
2	0.056475		0.01847497	4
3	0.042656		0.0072115	7
7	0.002813		0.00554697	3

rank	General Influence	APPROX-70	Observation Influence	rank
0	1		1	0
1	0.201726		0.12653681	1
5	0.119321		0.08627022	6
4	0.099498		0.08138394	2
6	0.098167		0.04369069	5
2	0.075422		0.01928228	4
3	0.060053		0.00976759	7
7	0.004032		0.00764253	3

Figure 5.47: Influence values of all nodes, sorted by influence, for pipeline no.3

Figure 5.48 provides the ranking of the nodes, as resulted by the computation of influence by following both the exact and the approximate approaches, and for

all sampling ratios used.

	EXACT	APPROX-10		APPROX-30		APPROX-50		APPROX-70	
Influence rank	General	General	Observation	General	Observation	General	Observation	General	Observation
1	0	0	0	0	0	0	0	0	0
2	1	1	2	1	1	1	1	1	1
3	5	6	1	5	6	5	6	6	6
4	4	4	6	6	2	6	2	5	2
5	6	3	5	4	5	4	5	4	5
6	2	5	4	2	4	2	4	2	4
7	3	2	3	3	7	3	7	3	7
8	7	7	7	7	3	7	3	7	3

Figure 5.48: Node ranking by influence for exact and approximate cases, for both general and given observation cases, for pipeline no.3

Figure 5.49 provides information about how much each approach verges on the ground truth provided above.

The first column (pos) includes in which positions of the ranking each approach's result match with the ground truth. As a result, the second column (hits) shows the amount of nodes found in the same ranking position with the ground truth, i.e., in which nodes' ranking do the ground truth and each approach agree.

The three last columns show how much each approach achieved to agree with the ground truth at the top 25% of the pipeline node, the top 40% and finally at the top influential node.

	Pipeline no.3				
	Position of hits	Hits(%nodes,position)	top-25%	top-40%	top-1
Heuristic-combo	0	0.125	0.5	0.33	1
Heuristic-Out degree	-	0	0	0	0
Heuristic-Distance	0	0.125	0.5	0.33	1
General-Exact	0,1	0.25	1	0.55	1
General-Approx-10%	0,1,2	0.4	1	0.67	1
Observation-Approx-10%	0,5	0.25	0.5	0.33	1
General-Approx-30%	0,1	0.25	1	0.55	1
Observation-Approx-30%	0,1,2,6,7	0.625	1	1	1
General-Approx-50%	0,1	0.25	1	0.55	1
Observation-Approx-50%	0,1,2,6,7	0.625	1	1	1
General-Approx-70%	0,1,2,3	0.5	1	1	1
Observation-Approx-70%	0,1,2,6,7	0.625	1	1	1

Figure 5.49: Figure of the quality of results for all cases, for pipeline no.3

They very first conclusion that can be drawn by figure 5.49 is that our worst performing approaches are better than the best performing heuristic approach. The heuristic approaches continue to have very low hit rate and in this pipeline, they even fail to find the second most influential node, or in the out_degree_heuristic, even the most influential node.

Our general case approach starts returning wrong results after about the top-30% of the nodes.

The above conclusions can be better understood by the diagrams in figures 5.50 and 5.51:

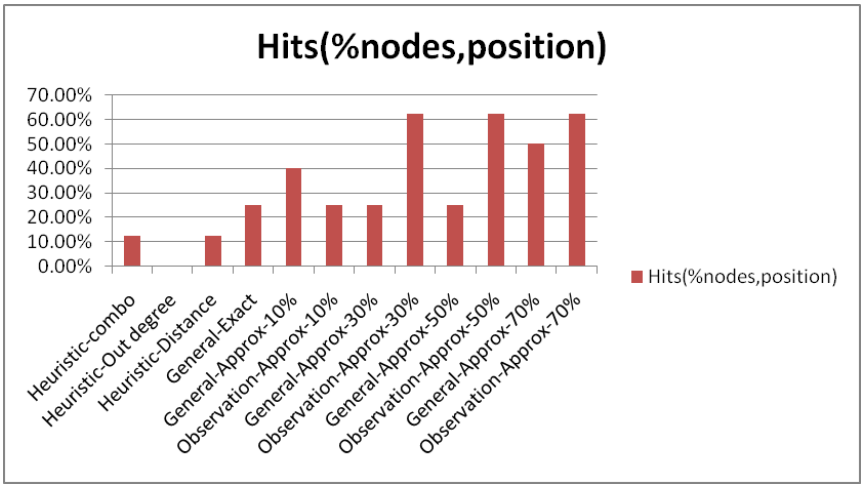


Figure 5.50: Hit rate for pipeline no.3

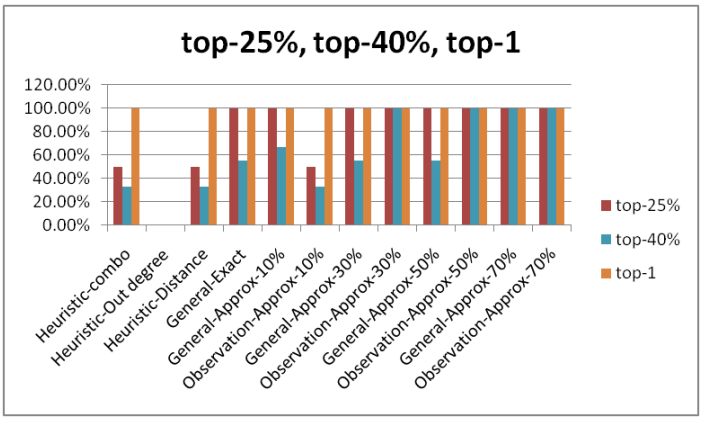


Figure 5.51: Top-25%, top-40% and top-1 results for pipeline no.3

The above figures explain better what we have earlier mentioned: observation-based approach performs better than the general case, and all our approaches, even when approximations are used, outperform the result quality of the heuristics.

We should point out that the general-approximate-10% method did remarkably well in terms of hits. However, we cannot accept this result as credible, and as a result we believe that in this case, the method was "lucky" enough to sample the most representative/important possible worlds for each node.

We continue by providing two diagrams that show the square distance from the ground truth, per ranking position totally, for all approaches

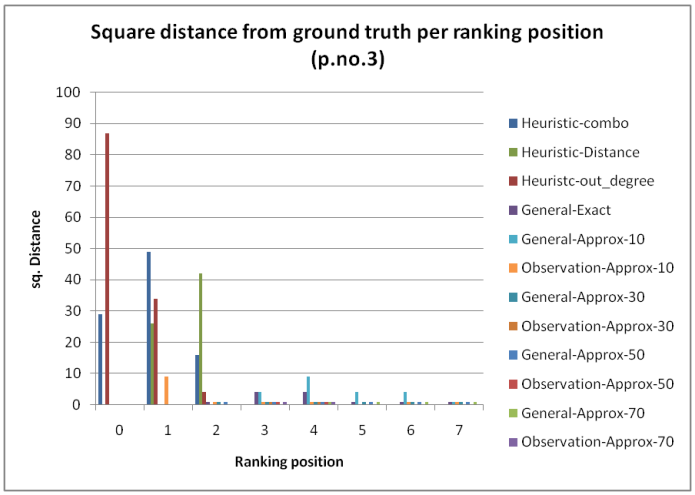


Figure 5.52: Square distance from ground truth per ranking position

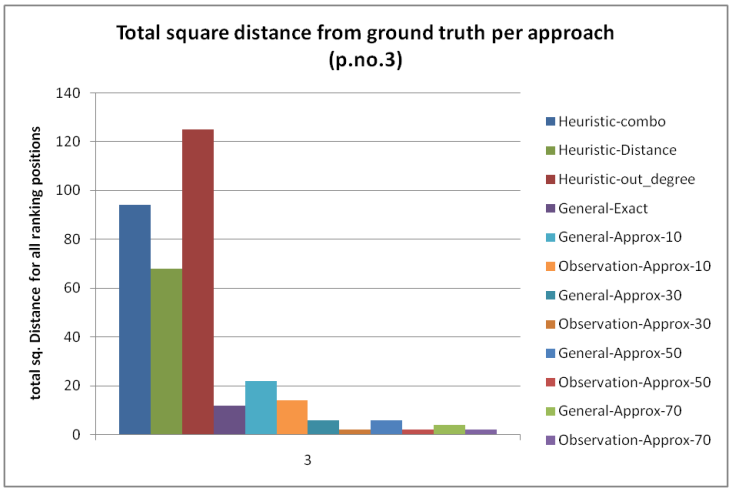


Figure 5.53: Total square distance from ground truth per approach

A remarkable spot of the above results is the following: if we take a closer look at the ranking that has been returned, we may notice that even in the cases where our approach did not do as well as we would expect, each wrongly ranked node is not more than one position away than it should be. As a result, we may say that misplaced nodes appear close to their original ranking and thus, should we be asked to return the top-4 influential nodes, what will most probably happen is to return the top-3 influential and the 5^{th} more influential node. Unfortunately, the above conclusion does not apply to all cases: the approximate-10% has not managed to rank the nodes close to their correct ranking position. However, even with this problem, it still performs better than the heuristics.

In this pipeline, our approach clearly outperforms the heuristic approaches, which not only misplaced most nodes in the ranking, but also the ranking was many positions away from the correct.

The following two figures (5.54 and 5.55) show each node's influence value convergence to the influence value by the ground truth, as calculated by each approximation approach.

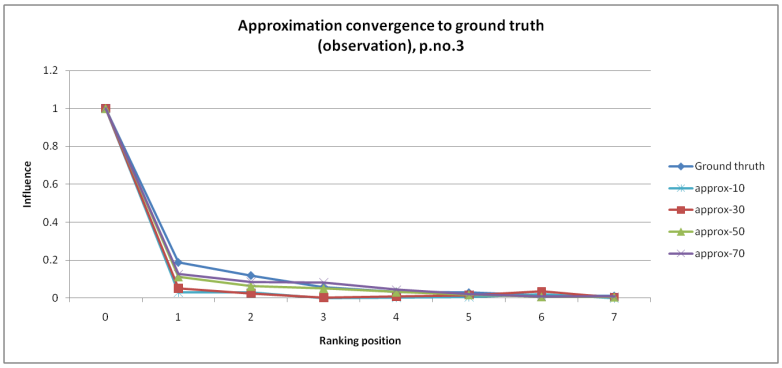


Figure 5.54: Influence value convergence to ground truth value for each approximation approach, observation-based approach

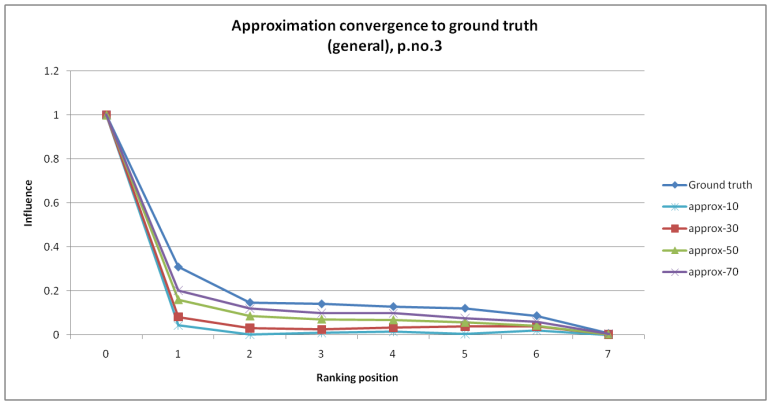


Figure 5.55: Influence value convergence to ground truth value for each approximation approach, general case

The results of the above figures (Fig. 5.54 and 5.55) show that in this case our approximation approaches perform very well. All methods converge to the ground truth results, increasing the sampling ratio increases the quality of the results.

The following figures (5.56 and 5.57) show how the influence value is correlated to each node's C_i and D_i sets size.

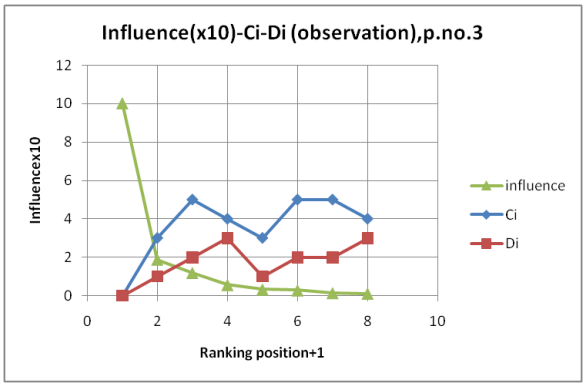


Figure 5.56: Influence value correlation to size of sets C_i and D_i, observation-based approach

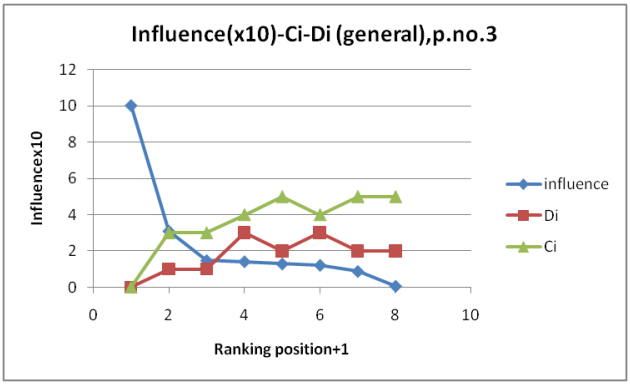


Figure 5.57: Influence value correlation to size of sets C_i and D_i, general case

In figure 5.57 we show how the influence value depends on the size of the set C_i of each node under consideration. In this case, this correlation is more clear, since the sizes of the C_i sets are bigger than in the two previous pipelines, whereas the size of the sets D_i are quite the same.

5.4.3.6 Concluding remarks

A very important conclusion that can be drawn by pipeline no.3 derives from the introduced 10% approximation method. This is the first pipeline where we have used this sampling ratio, as in pipeline no.3 the sizes of the C_i sets are larger than in the other two pipelines. The results of the 10% approximation method

show that our method is applicable to pipelines with larger C_i sets, since the quality of the returned results is quite good -better than the heuristics- and the execution time cost is proportional to the time cost by the heuristic approaches. Specifically, figures 5.50 and 5.51 show that even 10% sampling ratio is enough to have good quality results in pipelines with large C_i sets. As expected, 10% sampling ratio is slightly worse than the rest sampling ratios used. This, however, does not imply many "mis-ranked" nodes, but from the bigger distance of the few mis-ranked nodes from their correct ranking position, as showed in figures 5.52 and 5.53. The 10% approximation method has achieved good performance for the top-"few" nodes and it is surely much more preferable than any heuristic, since they cost the same in means of execution time.

Apart from the common conclusions to the previous two pipelines which we believe that have already been thoroughly discussed, another important conclusion drawn by pipeline no.3 is the following: even though the ranking of the influential nodes is not perfect by using our approximate approaches, the misplaced nodes are positioned very close to their original ranking position (see figures 5.52 and 5.53). Unlike the heuristic approaches which fail in this aspect, even our average performing 10% approximation method provided good results in terms of the square distance of each ranking position from the ground truth. This is a very important characteristic: if a user asks to have the top-k influential nodes refined, the worst-case scenario would be to have the top-(k-1)$\cup$(k+1)th ranked nodes refined, in most of the cases.

5.4.4 Comparison with the influence definition provided by [1]

We have already discussed in section 2.5 that the definition of influence provided by the paper of Re and Suciu [1] does not consider the case of uncertain intermediate operators and thus, all the system's uncertainty derives from the input data. We expect that, should their definition be applied to our problem, the results would be of very low quality, since only input data would be identified as influential components of the pipeline. As a result, there is no point comparing

their method to ours, since we not only seek for the influential input data, but for all pipeline influential operators and input data.

5.5 Analysis of the results, general remarks and discussion

In this section, we provide some general remarks on our approach, the heuristic metrics tested and the experimental results. We also sum up our conclusions for each approach used, and we propose the most appropriate method (e.g., exact or approximate) for each case.

5.5.1 Execution time

In general, our experiments have shown that the execution time for our approach in the exact cases is about 5-8 times more than the time needed by the heuristic approaches, and 1-5 times more for the approximate cases. This is definitely a significant increase in the execution time. However, a comparison of the quality of the returned results by our approaches and the heuristic approaches shows that the time overhead is clearly taken over by the results quality, in most of the cases. There are cases (e.g., pipeline 3, figure 5.13) where there is no point using the heuristic approaches at all, due to very low result quality, and others (e.g., pipeline 3, figure 5.33) where our approximation approach results outperform the heuristics, with only a small time overhead.

As a result, when choosing between these approaches and having only the execution time cost in mind, there might be a few cases where it is more beneficial to choose a heuristic approach, however paying the cost of missing even the top-1 or the top-3 influential operators correctly identified. Also, there are cases where the approximation approach performs remarkably well in almost the same time as the heuristics, where it is surely more beneficial to use our approach, since the quality of the results is much better than the quality of the heuristic approaches.

5.5.2 Heuristics

Overall, the quality of the returned results by the heuristic approaches is lower than our approaches in most of the cases, even when low sampling ratio (30%) is used.

Among the three heuristics used, the combo_heuristic returned the highest quality results in most of the cases. Also, the distance_heuristic returned good quality results in many cases. The lowest quality results where returned by the out_degree_heuristic, which failed to identify the top-k influential nodes, in most of the cases. However, it was "by definition" expected for the out_degree_heuristic to fail in identifying the most influential node, i.e., the root node, since the root node has zero out-degree.

5.5.3 Observation based approach

The experimental results for the observation based approach, both for the exact and the approximate cases, have shown that we can achieve high quality results with comparably low execution time, especially when approximations are used. We consider that this approach will be very helpful when deciding to include users in the pipeline repairing process: the observation based approach combined with user feedback (section 4.3) will boost the repairing process, enabling to come up with a much more beneficial pipeline improvement plan, since we will be able to know the influential operators of a specific execution and the desired output value of it.

5.5.4 General approach

The results presented for both the exact and the approximate cases for the general approach (no observed values provided) have shown that this approach is a little more time consuming - in terms of execution time - than the observation based approach. Due to the fact that we are not provided with observed output values coming from an execution (we study how the pipeline works in general), we have to consider some input values for all nodes of the set C_i, for every pipeline node v_i.

For each node, we also have to calculate its mean output false positive/negative probabilities.

The above indicate that it is expected, for the general approach, to be more time consuming and less accurate than the observation based approach. However, we have shown through our experiments that it can achieve almost the same quality of the results with just 10-20% more execution time and sampling ratio than the observation based approach. This is very important, as it enables us to reason about the influential operators of a pipeline, without having any information coming from an execution. Moreover, the general approach avoids being overfit by observations, ultimately allowing to have a more general view of how the operators affect the output of any pipeline.

5.5.5 Approximation

For the experimental evaluation of our approach we have used three, and in one case four, sampling ratios over possible worlds, for approximating the influence of each node: 30%, 50%, 70%, and 10% sampling ratios. Our experimental results have shown that even for 50% ratio -where the execution time is considerably low- we have achieved very good quality of the returned results. Especially for the observation based approach, our 50% approximation approach managed to return nearly perfect results in most of the cases. In the general case, high quality results were returned by using 70% sampling ratio.

In pipeline no.3, where also 10% sampling ratio has been used, the results are still much better than the results of the heuristic approaches, while the execution time is the same as the execution time needed by the heuristic approaches. This show the efficiency of our technique, that it can achieve high quality results while maintaining a low execution time cost.

The above conclusions show that our approach performs well even when approximation is used to reduce the execution time needed. As a result, by using approximation we manage to output high quality results while the execution time is comparable to the execution time of the heuristic approaches.

At this point we should make a very important notice about sampling in such a setting. We recognise that our approaches have not followed the common path

of using sampling ratios of at least one order of magnitude less than the exact case, in most of the cases, but even with 10% and 30% sampling ratios, we have achieved very low execution times, compared to the heuristics. However, in a setting where we look for possible worlds in which a node is the deciding vote, it is not feasible to sample too few possible worlds as it becomes very rare to sample a possible world where the deciding vote criterion holds and thus calculate the node's influence. Especially in our experiments, the small number of nodes of our pipelines make the use of smaller sampling ratios difficult.

We plan to extend the experimental evaluation in much bigger pipelines while considering even lower sampling ratios, in the near future. Early experiments have already shown that our approaches maintain the high quality of results as the ones that we have already presented.

5.5.6 Conclusions

Through our experiments, we have shown that our approach outperforms the heuristic approaches in all cases. The quality of our results has be proven to be much higher than the quality of the results returned by the heuristic approaches, and the execution time of our approach, when approximation is used, is comparable to the execution time of the heuristic approaches.

In general, it is more beneficial to use our approach even in the most simple pipelines. In larger and more complex pipelines, the use of our approach is imperative since the heuristic approaches fail to rank the influential nodes correctly.

Chapter 6

Future Work

This chapter proposes some possible future directions for enhancing and improving this work.

6.1 Reduce computational complexity

The problem we handle is an inherently hard problem with unavoidably high computational complexity. We plan to apply more optimizations in the future, in order to reduce the computational complexity. One step towards this goal has been made by introducing Monte Carlo sampling over the possible worlds.

6.1.1 Parallelization

In order to avoid large scale approximations that will reduce the results quality, we plan to process the pipeline in parallel, i.e. calculate possible worlds and multiple node's influence simultaneously, in order to achieve both high quality results and speed.

6.2 Find multiple causes simultaneously - cliques of influential nodes

We currently examine each node atomically. We should consider examining multiple nodes at a time in order to find out how *sets* of nodes influence the output.

6.3 Non boolean setting

Our current setting is Boolean, i.e. the random variables are assigned either True or False(0 /1). We intend to extend this work beyond Boolean values in the future.

6.4 Operator refinement

Although operator refinement simply translates to further targeted training to the selected node, we plan to provide a concrete approach to this problem in the future. This because much more study is needed, in order to model the process and examine the case where not only the probabilities, but also the operator's observed output values, are changed after the refinement has taken place.

6.5 User feedback

We leave for future work the development of a technique to exploit the user feedback, when such precious information is available.

Chapter 7

Conclusions

Enhancing information extraction pipelines with operators from the Machine Learning community helps Community Information Management (CIM) platforms adapt to more domains with minimum human involvement and improve over time. However, such operators are inherently uncertain and thus add another source of uncertainty besides the input data.

In this thesis, we have investigated a novel lineage problem setting, where extraction pipelines are populated with uncertain operators that come from the Machine Learning community. Moreover, we have proposed a method for quantifying the influence of each uncertain operator to the returned results and use this information to efficiently repair pipelines populated with uncertain operators. Providing a low-cost repairing strategy allows to rapidly improve the system by only refining the top-k influential operators. Moreover, we have considered approximation techniques which enabled us to achieve equally high quality results, as in the exact case, while cutting down on the execution time cost.

We have also provided a generalization of the influence definition of Ré and Suciu [1], in order to handle uncertain operators. We strongly believe that the generalization of their definition introduces new perspectives in the intersection of the areas of information extraction and uncertain data management. The intuition behind this suggests that our approach enables the study of more complex and sophisticated pipelines, extends its use to a wider range of research areas and reveals a great amount of challenges for the future.

Our results have shown that our approach outperforms all heuristic approaches tested. Up to our knowledge there is no prior art handling an equivalent problem, ergo there are unfortunately no experimental results to compare with.

Concluding, we believe that this thesis provides the first steps towards efficient lineage processing in uncertain operator pipelines, influence analysis, and beneficial refinement plan of uncertain operators in such a setting.

Chapter 8

References

[1] C.Re, D.Suciu "Approximate Lineage for Probabilistic Databases", *In VLDB 2008*

[2] H.Chockler, J.Halpern, "Responsibility and Blame: A Structural-Model Approach", *In Journal of Artificial Intelligence Research, 2004*

[3] A.D.Sarma, A.Jain, P.Bohannon, "PROBER: Ad-Hoc Debugging of Information Extraction Pipelines", *Technical Report, April 2010*

[4] P.Bohannon, S.Merugu, C.Yu, V.Agarwal, P.DeRose, A.Iyer, A.Jain, V.Kakade, M.Muralidharan, R.Ramakrishnan, W.Shen, "Purple SOX Extraction Management System", *In ACM SIGMOD Record 37(4), 2008*

[5] A.Meliou, W.Gatterbauer, K.Moore, D.Suciu, "*WHY SO? or WHY NO?* Functional Causality for Explaining Query Answers", *In MUD Workshop, VLDB 2010*

[6] A.Deshpande, L.Getoor P.Sen, "Graphical Models for Uncertain Data", *In Managing and Mining Uncertain Data, 2009 - Springer, book chapter*

[7] R.Ikeda, J.Widom, "Data Lineage: A Survey", *In Stanford InfoLab, Technical Report, 2009*

[8] A.Meliou, W.Gatterbauer, J.Halpern, C.Koch, K.Moore, D.Suciu, "Causality in Databases", *In IEEE Data Engineering Bulletin, Special Issue on Provenance, 33(3), Sept. 2010*

[9] A.Meliou, W.Gatterbauer, K.Moore, D.Suciu, "The Complexity of Causality and Responsibility for Query Answers and non-Answers", *To appear in PVLDB 2011*

[10] C.Hitchcock, "Probabilistic Causation", entry in the *Stanford Encyclopedia of Philosophy,* http://plato.stanford.edu/entries/causation-probabilistic/

[11] J.Y.Halpern, J.Pearl, "Causes and Explanations: A Structural-Model Approach. Part I: Causes / Part II: Explanations", *British Journal for the Philosophy of Science, December 2005*

[12] P.Agrawal, A.Das Sarma, J.Ullman, J.Widom, "Foundations of Uncertain-Data Integration", *In VLDB 2010*

[13] J.Pearl, "Causality: Models, Reasonging and Inference, 2^{nd} edition", *Cambridge University Press*

[14] A.Doan, R.Ramakrishnan, F.Chen, P.DeRose, Y.Lee, R.McCann, M.Sayyadian, W.Shen, "Community Information Management", *In IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases, 29(1), March 2006*

[15] P.DeRose, W.Shen, F.Chen, Y.Lee, D.Burdick, A.Doan, R.Ramakrishnan, "DBLife: A Community Information Management Platform for the Database Research Community", *In CIDR-07 (demo)*

[16] D.Suciu, "Probabilistic Databases", *In: Encyclopedia of Database Systems (2009), p.2150-2155*

[17] N.Dalvi, C.Re, D.Suciu, "Probabilistic Databases: Diamonds in the Dirt", *In CACM, vol.52, no.7, 2009*

[18] T.Heinis, G.Alonso, "Efficient lineage tracking for scientific workflows", *In SIGMOD 2008, Proceedings of the 2008 ACM SIGMOD international conference on Management of data, p.1007-1018*

[19] N.Dalvi, D.Suciu, "Management of Probabilistic Data, Foundations and Challenges", *In PODS 2007*

[20] N.Dalvi, D.Suciu, "Efficient Query Evaluation on Probabilistic Databases", *In VLDB 2004*

[21] B.Kanagal, J.Li, A.Deshpande, "Sensitivity Analysis and Explanations for Robust Query Evaluation in Probabilistic Databases", *In SIGMOD 2011*

[22] C.Re, N.Dalvi, D.Suciu, "Efficient Top-k Query Evaluation on Probabilistic Data", *In ICDE 2007*

[23] A.D.Sarma, M.Theobald, J.Widom, "Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases", *In ICDE 2008*

[24] O.Benjelloun, A.D.Sarma, A.Halevy, J.Widom, "ULDBs: Databases with

Uncertainty and Lineage", *In VLDB 2006*
[25] P.Sen, A.Deshpande, L.Getoor, "Representing Tuple and Attribute Uncertainty in Probabilistic Databases", *In ICDM Workshop on Data Mining of Uncertain Data (DUNE), 2007*
[26] A.Gupta, "Data Provenance", *In: Encyclopedia of Database Systems (2009), p.608*