Book Title
Book Editors
IOS Press, 2003

Belief Propagation on Uncertain Schema Mappings in Peer Data Management Systems [1]

Philippe Cudré-Mauroux [2], and Karl Aberer
School Of Computer and Communication Sciences
EPFL – Switzerland

Abstract. Until recently, most data integration techniques involved central components, e.g., global schemas, to enable transparent access to heterogeneous databases. Today, however, with the democratization of tools facilitating knowledge elicitation in machine-processable formats, one cannot rely on global, centralized schemas anymore as knowledge creation and consumption are getting more and more dynamic and decentralized. Peer Data Management Systems (PDMS) provide an answer to this problem by eliminating the central semantic component and considering instead compositions of local, pair-wise mappings to propagate queries from one database to the others.

In the following, we give an overview of various PDMS approaches; all the approaches proposed so far make the implicit assumption that all schema mappings used to reformulate a query are correct. This obviously cannot be taken as granted in typical PDMS settings where mappings can be created (semi) automatically by independent parties. Thus, we propose a totally decentralized, efficient message passing scheme to automatically detect erroneous schema mappings in a PDMS. Our scheme is based on a probabilistic model where we take advantage of transitive closures of mapping operations to confront local belief on the correctness of a mapping against evidences gathered around the network. We show that our scheme can be efficiently embedded in any PDMS and provide an evaluation of our techniques on large sets of automatically-generated schemas.

Keywords. Data Integration, Peer Data Management Systems, Belief Propagation

1. Introduction

Data integration techniques have traditionally revolved around global schemas to query heterogeneous databases in a transparent way: popular techniques such as LAV (Local-

[1]The work presented in this chapter was carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the IST European project Evergrow No 001935, and was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

[2]Correspondence to: Philippe Cudré-Mauroux, School Of Computer and Communication Sciences, Station 14, EPFL, 1010 Lausanne, Switzerland. Tel.: +41 21 693 6787; Fax: +41 21 693 8115; E-mail: philippe.cudre-mauroux@epfl.ch.

As-View) or GAV (Global-As-View) drew considerable attention as they permit deterministic reformulations of queries over various data sources. These centralized approaches, however, require the definition of an integrated schema supposedly subsuming all local schemas. This requirement is particularly stringent in highly heterogeneous, dynamic and decentralized environments such as the Web or Peer-to-Peer overlay networks. In these settings, largely autonomous and heterogeneous data sources coexist without any central coordination. Keeping a global schema up-to-date or defining a schema general enough to encompass all information sources is thus unmanageable in practice in this context. This evolution motivated the research community to imagine new paradigms for handling heterogeneous data in decentralized environments. In this chapter, we focus on the Peer Data Management Systems (PDMS) approach, which recently drew considerable attention in this context [1,2].

PDMS take advantage of local, pairwise schema mappings between pairs of databases to propagate queries posed against a local schema to the rest of the network in an iterative and cooperative manner. No global semantic coordination is needed as peers only need to define local mappings to a small set of related databases to be part of the global network of databases. Once a database sends a query to its immediate neighbors through local mapping links, its neighbors (after processing the query) in turn propagate the query to their own neighbors, and so on and so forth until the query reaches all (or a predefined number of) databases. The way the query spreads around the network mimics the way messages are routed in a Peer-to-Peer (P2P) system, thus the appellation *Peer Data Management Systems*.

The vast majority of PDMS approaches, however, propagate queries without concern on the validity or quality of the mappings. This obviously represents a severe limitation, as one cannot expect any level of consistency or quality on PDMS mappings for several reasons: first, remember that PDMS target large-scale, decentralized and heterogeneous environments where autonomous parties have full control on schema designs. As a result, we can expect irreconcilable differences on conceptualizations (e.g., epistemic or metaphysical differences on a given modelization problem, see also [3]) among the databases. Also, the limited expressivity of the mappings, usually defined as queries or using an ontology definition language like OWL [4], precludes the creation of correct mappings in many situations (e.g., mapping an attribute onto a relation). Finally, given the vibrant activity on (semi-) automatic alignment techniques (see [5] for a recent overview), we can expect some (most?) of the mappings to be generated automatically in large-scale settings, with all the evident quality problems associated.

In the following, we propose a probabilistic technique to determine the quality of schema mappings in PDMS settings in a totally automated way and without any form of central coordination. As peers do not always share common data, mapping errors are typically difficult to discover from a local perspective in PDMS. Our methods are based on the analysis of cycles and parallels paths in the graph of schema mappings: after detecting mapping inconsistencies by comparing transitive closures of mapping operations, we build a global probabilistic inference model spanning the entire PDMS system. We show how to construct this model in a totally decentralized way by involving local information only. We describe a decentralized and efficient method to derive mapping quality measures from our model. Our approach is based on a decentralized version of iterative sum-product message passing techniques (loopy belief propagation). We show how to embed our approach in PDMS systems with a very modest communication overhead by

piggybacking on normal query processing operations. Finally, we present an evaluation of our techniques applied on large corpuses of automatically-generated schemas.

2. Peer Data Management Systems

Integration systems traditionally adopted the Federated Databases model (see Figure 1 a). This model provides a uniform access to multiple heterogeneous data sources through a centralized Federated Database Server. Wrappers deal with syntactic heterogeneity by converting each local database to a common representation. In this way, data can be stored using various data models (e.g., semi-structured, text files) at the local databases, while the Federated Database only deals with a unique – typically relational – representation. A mediator integrates data with the same meaning but stored using different schemas. It can for example define correspondences (mappings) between various schemas to reformulate a query posed against the integrated schema to queries posed against each local database. The Local As View (LAV) approach defines each local database as a view on an federated, centralized schema. In the Global As View approach, on the other hand, the federated schema is defined in terms of the local schemas. Clients can thus query the Federated Database Server, typically through a SQL interface, which is able to reformulate the query thanks to the mediator, query the local databases and collect results through the wrappers to finally return all answers to the clients.

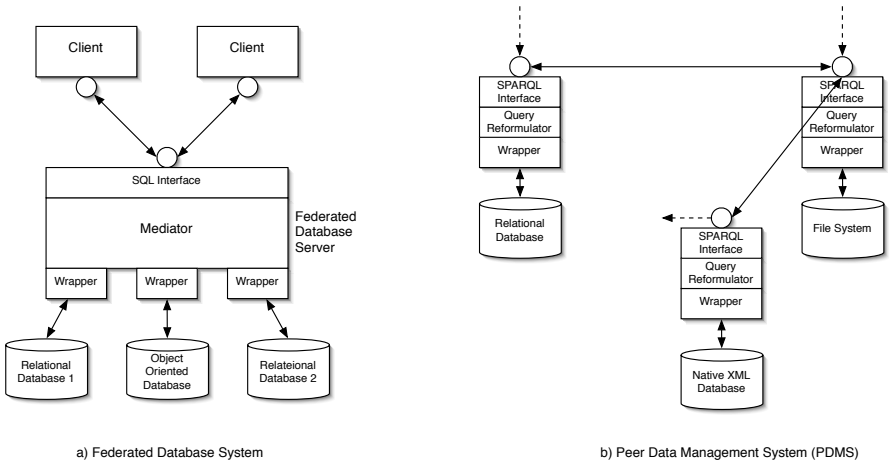

Figure 1. Two data integration models: Federated Databases (a) based on a central federated schema and Peer Data Management Systems (b) based on peer-to-peer mappings between the databases

.

One of the main limitations of this model lies in its centralization: the Federated Database Server represents a single point of failure and is responsible for integrating *all* databases taking advantage of a *global* schema. The server severely hampers the scalability of the approach and violates the autonomy of the local sources. Aware of these limitations, the research community recently explored new techniques to manage and integrate

data in large scale, decentralized or Peer-to-Peer settings (see [1] for a recent tutorial on Semantic Overlay Networks). In Peer Data Management Systems settings (see Figure 1 b), the centralized mediator disappears and is replaced by an unstructured collection of peer-to-peer mappings between the databases. PDMS take advantage of these local, pairwise schema mappings between pairs of databases to propagate queries posed against a local schema to the rest of the network in an iterative and cooperative manner. No global semantic coordination is needed as peers only need to define local mappings to a small set of related databases to be part of the global network of databases. Once a database sends a query to its immediate neighbors, its neighbors (after processing the query) can reformulate the query against their local databases, and propagate the query to their own neighbors, and so on and so forth until the query reaches all (or a predefined number of) databases.

Mappings in a PDMS can be defined as views, or by taking advantage of an ontology language such as OWL. Note that in practice, the way the various schemas and schema mappings are organized might be independent of the physical organization of the databases: Figure 2 depicts a Semantic Overlay Network where the organization of the peer at the overlay layer is uncorrelated with the organization of the schemas and schema mappings at the mediation layer. In this way, peers can create or share schemas irrespective of their location in the overlay network.

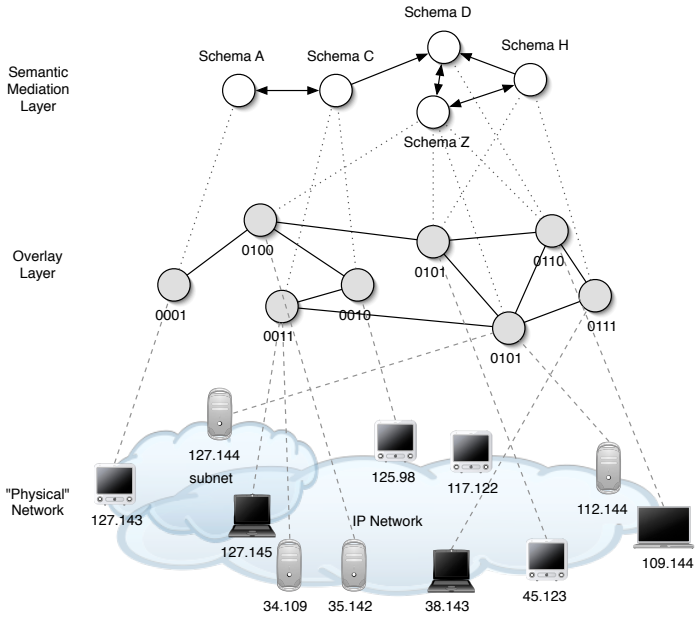

Figure 2. A three-layered representation of a Semantic Overlay Network: machines (bottom layer) organize themselves into a peer-to-peer overlay (intermediate layer) and communicate through a Semantic Mediation Layer (top layer) defining mappings between the various schemas used in the network

.

In the following, we concentrate on a probabilistic analysis of the Semantic Mediation Layer in order to detect inconsistent schema mappings. Our approach is quite natu-

rally based on some of our previous ideas [6,7] for analyzing the network of mappings. Others have also focused on PDMS settings recently: in [8], Bernstein et al. formulated requirements for P2P databases in terms of local mappings and semi-automatic solutions for data integration. Edutella [9], based on a super-peer topology, was one of the early PDMS systems deployed. Piazza [10,11] is a PDMS system which provides efficient query reformulations algorithms for relational and semi-structured data models. The Hyperion [12] project relies on rules to propagate data in a PDMS network according to mapping tables. PeerDB [13] examines the problem of sharing relational data in P2P networks from an information retrieval perspective. One of our earlier systems, GridVine [14], is based on a structured overlay layers and thus supports three uncorrelated layers as depicted in Figure 2.

3. Problem Definition

For the sake of clarity, we consider in the following only simple PDMS settings where each peer represents a separate database (i.e., there is a direct correlation between the overlay layer and the semantic mediation layer). Our methods work in an identical manner in more complex settings.

We model PDMS as collections of peers; Each individual peer p represents a database storing data according to a distinct structured schema. As we wish to present an approach as generic as possible, we do not make any assumption on the exact data model used by the databases in the following but illustrate some of our claims with examples in XML and RDF. We only require the databases to store information with respect to some concepts we call *attributes* a (e.g., attributes in a relational schema, elements or attributes in XML and classes or properties in RDF). Additionally, we suppose that each peer can be identified and contacted by a unique identifier (e.g., an IP address or a peer ID in a P2P network).

Peers are connected one another through (un)directed edges representing (un)directed pairwise schema mappings. A schema mapping m allows a query posed against a local schema to be evaluated against another schema. This operation is uni or bi-directional depending on the language used to express the mappings. Again, we do not make assumptions on that language, except that it should allow to connect pairs of semantically similar attributes. Note that this language may for example allow for syntactic transformations (e.g., transforming a date from one format to another) or complex mappings (e.g., mapping an attribute to part of another attribute). Finally, remember that our fundamental assumption is that some mappings might be incorrect, i.e., mappings might map an attribute from one database to a semantically irrelevant attribute in another database.

We consider queries posed locally against the schema of a given peer. Queries are composed of generic selection / projection operations *op* on attributes. For each attribute a_i appearing in the query, the system (or the expert user) defines a semantic threshold θ_{a_i} under which the query should not be propagated any further: as the query gets propagated throughout the network of peer databases, each intermediate peer checks the probability $P(a_i = correct)$ of each attribute in the query being semantically preserved by a mapping operation. The query is forwarded through a mapping link to another peer database only if all attributes are preserved, that is if for all a_i, $P(a_i = correct) > \theta_{a_i}$ for the given mapping. Note that other per-hop forwarding behaviors could easily be implemented with our techniques (see [7]) but we stick to the given scheme for simplicity.

Given this setting, our goal is to provide probabilistic guarantees on the correctness of a mapping operation, i.e., to determine $P(a_i = correct)$. As any processing in a PDMS, we wish our methods to operate without any global coordination in a purely decentralized manner. Also, we would like our methods to be totally automated, as precise as possible and fast enough to be applied on large schemas or ontologies.

3.1. An Introductory Example

Before delving into technicalities, we start with a high-level, introductory example of our approach. Let us consider the simple PDMS network depicted in Figure 3. This network is composed of four databases $p_1, \ldots, p_4$. All databases store a collection of XML documents related to pieces of art, but structured according to four different schemas (one per database). Each database supports XQuery as query language. Various pairwise XQuery schema mappings have been created (both manually and automatically) to link the databases; Figure 4 below shows an example of mapping between two schemas and how one can take advantage of this mapping to resolve a query posed against two different schemas.

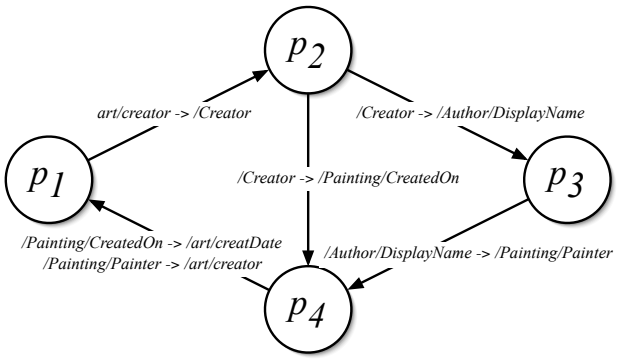

Figure 3. A simple directed PDMS network of four peers and five schema mappings, here depicted for the attribute *Creator*

Let us suppose that a user in p_2 wishes to retrieve the names of all artists having created a piece of work related to some river. The user could locally pose an XQuery like the following:

```
q_1 =
FOR $c IN distinct-values (ArtDatabank//Creator)
WHERE $c/..//Item LIKE "%river%"
RETURN <myArtist> $c </myArtist>
```

This query basically boils down to a projection on the attribute *Creator*: $op_1 = \pi_{Creator}$ and a selection on the title: $op_2 = \sigma_{Item=\%river\%}$. The user issues the query and patiently awaits for answers, both from his local database and the rest of the network.

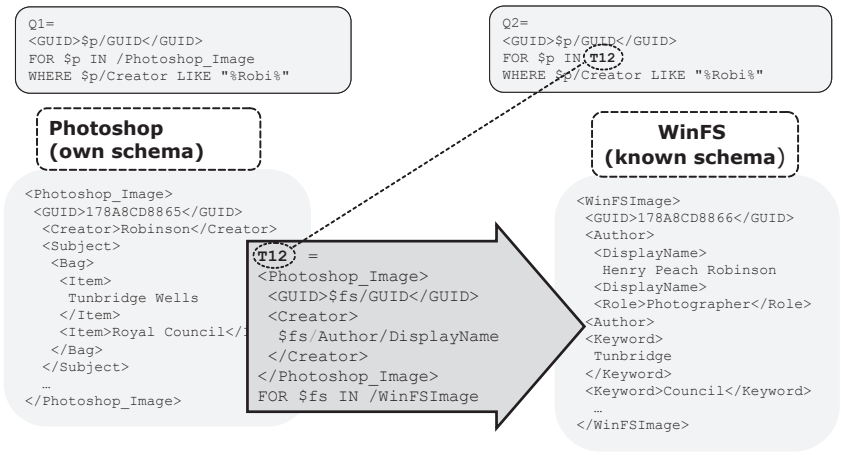

Figure 4. An example of schema mapping (here between peers p_2 and p_3) expressed as an XQuery

In a standard PDMS, the query would be forwarded through both outgoing mappings of p_2, generating a fair proportion of false positives as one of these two mappings (the one between p_2 and p_4) is incorrect for the attribute $Creator$ (the mapping erroneously maps *Creator* in p_2 onto *CreatedOn* in p_4, see Figure 3). Luckily for our user, the PDMS system he is using implements our belief propagation techniques. Without any prior information on the mappings, the system detects inconsistencies for the mappings on *Creator* by analyzing the cycles $p_1 \rightarrow p_2 \rightarrow p_4 \rightarrow p_1$ and $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4 \rightarrow p_1$, as well as the parallel paths $p_2 \rightarrow p_4$ and $p_2 \rightarrow p_3 \rightarrow p_4$ in the mapping network. In a decentralized process, the PDMS constructs a probabilistic network and determines that the semantics of the attribute *Creator* will most likely be preserved by all mappings, except by the mapping between p_2 and p_4 which is more likely to be faulty. Thus, this specific query will be routed through mapping $p_2 \rightarrow p_3$, and then iteratively to p_4 and p_1. In the end, the user will retrieve all artist names as specified, without any false-positive since the mapping $p_2 \rightarrow p_4$ was ignored in the query resolution process.

4. Modeling PDMS as Factor-Graphs

In the following, we take advantage of query messages being forwarded from one peer to another to detect inconsistencies in the network of mappings. We represent individual mappings and network information as related random variables in a probabilistic graphical model. We will then efficiently evaluate marginal probabilities, i.e., mapping quality, using these models.

4.1. A Quick Reminder on Factor-Graphs and Message Passing Schemes

We give below a brief overview of message passing techniques. For a more in-depth coverage, we refer the interested reader to one of the many overviews on this domain, such as [15]. Note that Belief Propagation as introduced by Judea Pearl [16] is actually a specialized case of a standard message passing sum-product algorithm.

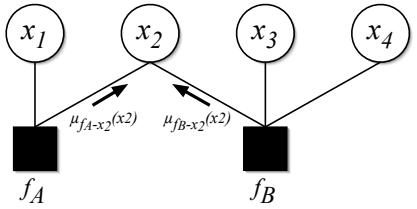

Figure 5. A simple factor-graph of four variables and two factors

Probabilistic graphical models are a marriage between probability theory and graph theory. In many situations, one can deal with a complicated global problem by viewing it as a factorization of several local functions, each depending on a subset of the variables appearing in the global problem. As an example, suppose that a global function $g(x_1, x_2, x_3, x_4)$ factors into a product of two local functions f_A and f_B: $g(x_1, x_2, x_3, x_4) = f_A(x_1, x_2) f_B(x_2, x_3, x_4)$. This factorization can be represented in a graphical form by the *factor-graph* depicted in Figure 5, where variables (circles) are linked to their respective factors (black squares). Often, one is interested in computing a *marginal* of this global function, e.g.,

$$g_2(x_2) = \sum_{x_1} \sum_{x_3} \sum_{x_4} g(x_1, x_2, x_3, x_4) = \sum_{\sim\{x_2\}} g(x_1, x_2, x_3, x_4)$$

where we introduce the summary operator $\sum_{\sim\{x_i\}}$ to sum over all variables but x_i. Such marginals can be derived in an efficient way by a series of *sum-product* operations on the local function, such as:

$$g_2(x_2) = \left(\sum_{x_1} f_A(x_1, x_2) \right) \left(\sum_{x_3} \sum_{x_4} f_B(x_2, x_3, x_4) \right).$$

Interestingly, the above computation can be seen as the product of two messages $\mu_{f_A \to x_2}(x_2)$ and $\mu_{f_B \to x_2}(x_2)$ sent respectively by f_A and f_B to x_2 (see Figure 5). The *sum-product* algorithm exploits this observation to compute all marginal functions of a factor-graph in a concurrent and efficient manner. Message passing algorithms traditionally compute marginals by sending two messages — one in each direction — for every edge in the factor-graph:

variable x to local factor f:

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$

local factor f to variable x

$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(f(X) \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

where $n(\cdot)$ stands for the neighbors of a variable / function node in the graph.

These computations are known to be exact for cycle-free factor-graphs; in contrast, applications of the sum-product algorithm in a factor-graph with cycles only result in approximate computations for the marginals [17]. However, some of the most exciting applications of the sum-product algorithms (e.g., decoding of turbo or LDPC codes) arise precisely in such situations. We show below that this is also the case for factor-graphs modelling Peer Data Management Systems.

4.2. On Factor-Graphs in Undirected PDMS

In the following, we explain how to model a network of mapping as a factor-graph. These factor-graphs will in turn be used in Section 5 to derive quality measures for the various mappings in the network.

4.2.1. Cyclic Mappings

Semantic overlay network topologies are not generated at random. On the contrary, they are constructed by (computerized or human) agents aiming at interconnecting partially overlapping information sources. We can expect very high clustering coefficients in these networks, since similar sources will tend to bond together and create cluster of sources. As an example, a study of an online network of related biologic schemas (in the the SRS system, http://www.lionbioscience.com) shows an exponential degree distribution and an unusually high clustering coefficient of 0.54 (as of May 2005). Consequently, we can expect semantic schema graphs to exhibit *scale-free* properties and an unusually high number of loops [18].

Let us assume we have detected a cycle of mappings $m_0, m_1, \ldots, m_{n-1}$ connecting n peers $p_0, p_1, \ldots, p_{(n-1)}, p_0$ in a circle. Cycles of mappings can be easily discovered by the peers in the PDMS network, either by proactively flooding their neighborhood with probe messages with a certain Time-To-Live (TTL) or by examining the trace of routed queries in the network. We take advantage of transitive closures of mapping operations in the cycle to compare a query q posed against the schema of p_0 to the corresponding query q' forwarded through all n mappings along the cycle: $q' = m_{n-1}(m_{n-2}(\ldots(m_0)(q))))$. q and q' can be compared on an equal basis since they are both expressed in terms of the schema of p_0. In an ideal world, $q' \equiv q$ since the transformed query q' is the result of n identity mappings applied on the original query q. In a distributed setting, however, this might not always be the case, both because of the lack of expressiveness of the mappings and of the fact that mappings can be created in (semi-) automatic ways.

When comparing an attribute a_i in an operation $op_q(a_i)$ appearing in the original query q to the attribute a_j from the corresponding operation $op'(a_j)$ in the transformed query q', three subcases may occur in practice:

$a_j = a_i$: this occurs when the attribute, after having been transformed n times through the mappings, still maps to the original attribute when returning to the semantic domain of p_0. Since this indicates a high level of semantic agreement along the cycle for this particular attribute, we say that this represents positive feedback f^+ on the mappings constituting the cycles.

$a_j \neq a_i$: this occurs when the attribute, after having been transformed n times through the mappings, maps to a different attribute when returning to the semantic domain of p_0. As this indicates some disagreement on the semantics of a_i along the cycle of mappings, we say that this represents negative feedback f^- on the mappings constituting the cycles.

$a_j = \perp$: this occurs when some intermediary schema does not have a representation for the attribute in question, i.e., cannot map the attribute onto one of its own attributes. This does not give us any additional (feedback) information on the level of semantic agreement along the cycle, but can still represent some valuable information in other contexts, for example when analyzing query forwarding on a syntactic level (see also [7]). In the current case, we consider that the probability on the correctness of a mapping drops to zero for a specific attribute if the mapping does not provide any mapping for the attribute.

We focus here on single-attribute operations for simplicity, but our results can be extended to multi-attribute operations as well.

Also, we take into account the fact that series of erroneous mappings on a_i can accidentally $compensate$ their respective errors and actually create a correct composite mapping $m_{n-1} \circ m_{n-2} \ldots \circ m_0$ in the end. Assuming a probability Δ of two or more mapping errors being compensated along a cycle in this way, we can determine the conditional probability of a cycle producing positive feedback $f^+_{\circlearrowleft}$ given the correctness of its constituting mappings $m_0, \ldots, m_{n-1}$:

$$P(f^+_{\circlearrowleft} | m_0, \ldots, m_{n-1}) = \begin{cases} 1 & \text{if all mappings correct} \\ 0 & \text{if one mapping incorrect} \\ \Delta & \text{if two or more} \\ & \text{mappings incorrect} \end{cases}$$

This conditional probability function allows us to create a factor-graph from a network of interconnected mappings. We create a global factor-graph as follows:

```
for all mapping m in PDMS
   add m.factor to global factor-graph;
   add m.variable to m.factor;
for all mapping cycle c in PDMS
   add c.feedback.factor to global factor-graph;
   add c.feedback.variable to c.feedback.factor;
   for all mapping m in mapping cycle c
      link c.feedback.factor to m.variable;
```

Figure 6 illustrates the derivation of a factor-graph from a simple semantic network of four peers $p_1, \ldots, p_4$ (left-hand side of Figure 6). The peers are interconnected through five mappings $m_{12}, m_{23}, m_{34}, m_{41}$ and m_{24}. One may attempt to obtain feedback from three different mapping cycles in this network:

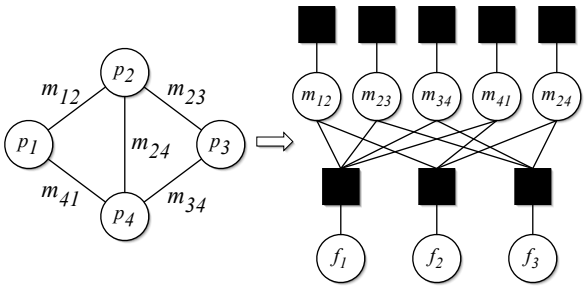

Figure 6. Modeling an undirected network of mappings as a factor-graph

$$\boldsymbol{f^1_{\circlearrowleft}} : m_{12} - m_{23} - m_{34} - m_{41}$$
$$\boldsymbol{f^2_{\circlearrowleft}} : m_{12} - m_{24} - m_{41}$$
$$\boldsymbol{f^3_{\circlearrowleft}} : m_{23} - m_{34} - m_{24}.$$

The right-hand side of Figure 6 depicts the resulting factor-graph, containing from top to bottom: five one-variable factors for the prior probability functions on the mappings, five mappings variables m_{ij}, three factors linking feedback variables to mapping variables through conditional probability functions (defined as explained above), and finally three feedback variables f_k. Note that feedback variables are usually not independent: two feedback variables are correlated as soon as the two mapping cycles they represent have at least one mapping in common (e.g., in Figure 6, where all three feedbacks are correlated).

4.3. On Factor-Graphs in Directed PDMS

One may derive similar factor-graphs in directed PDMS networks, focusing this time on directed mapping cycles and parallel mapping paths. Parallel mapping paths occur when two different series of mappings m' and m'' share the same source and destination. The conditional probability function for receiving positive feedback $f^+_{\rightrightarrows}$ through two parallel paths m' and m'' is as follows (see [19] for details):

$$P(f^+_{\rightrightarrows}|\{m'\},\{m''\}) = \begin{cases} 1 & \text{if all mappings correct} \\ 0 & \text{if one mapping incorrect} \\ \Delta & \text{if two or more} \\ & \text{mappings incorrect} \end{cases}$$

Figure 7 shows an example of a directed mapping network with four peers and six mappings. Feedback from two directed cycles and three pairs of parallel paths might be gathered from the network:

$$\boldsymbol{f^1_{\circlearrowleft}} : m_{12} \rightarrow m_{23} \rightarrow m_{34} \rightarrow m_{41}$$
$$\boldsymbol{f^2_{\circlearrowleft}} : m_{12} \rightarrow m_{24} \rightarrow m_{41}$$
$$\boldsymbol{f^3_{\rightrightarrows}} : m_{21} \| m_{24} \rightarrow m_{41}$$
$$\boldsymbol{f^4_{\rightrightarrows}} : m_{24} \| m_{23} \rightarrow m_{34}$$
$$\boldsymbol{f^5_{\rightrightarrows}} : m_{21} \| m_{23} \rightarrow m_{34} \rightarrow m_{41}.$$

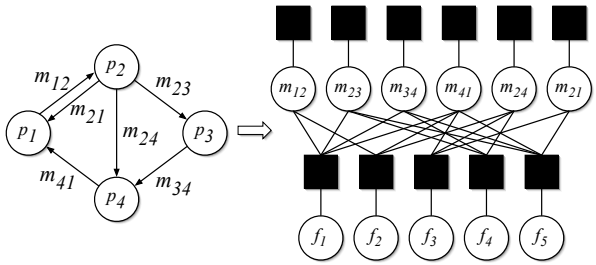

Figure 7. Modeling a directed network of mappings as a factor-graph

As for the undirected case, the right-hand side of Figure 7 represents the factor-graph derived from the directed mapping network of the left-hand side.

Since undirected mapping networks and directed mapping networks result in structurally similar factor-graphs in the end, we treat them on the same basis in the following. We only include two versions of our derivations when some noticeable difference between the undirected and the directed case surfaces.

5. Embedded Message Passing

So far, we have developed a graphical probabilistic model capturing the relations between mappings and network feedback in a PDMS. To take advantage of these models, one would have to gather *all* information pertaining to *all* mappings, cycles and parallel paths in a system. However, adopting this centralized approach makes no sense in our context, as PDMS were precisely invented to avoid such centralization. Instead, we devise below a method to embed message passing into normal operations of a Peer Data Management System. Thus, we are able to get globally consistent mapping quality measures in a scalable, decentralized and efficient manner while respecting the autonomy of the peers.

Looking back at the factor-graphs introduced in Section 4.2 and 4.3, we make two observations: i) some (but not all) nodes appearing in the factor-graphs can be mapped back onto the original PDMS graph, and ii) the factor-graphs contain cycles.

5.1. On Feedback Variables in PDMS Factor-Graphs

Going through one of the figures representing a PDMS factor-graph from top to bottom, one may identify four different kinds of nodes: factors for the prior probability functions on the mappings, variable nodes for the correctness of the mappings, factors for the probability functions linking mapping and feedback variables, and finally variable nodes for the feedback information. Going one step further, one can make a distinction between nodes representing local information, i.e., mapping factors and mapping variables, and nodes pertaining to global information, i.e., feedback factors and feedback variables.

Mapping back local information nodes onto the PDMS is easy, as only the nodes from which a mapping is departing need to store information about that mapping (see per hop routing behavior in Section 3). Luckily, we can also map the other nodes rather easily, as they either contain global but *static* information (density function in feedback factors), or information gathered around the local *neighborhood* of a node (Δ, observed

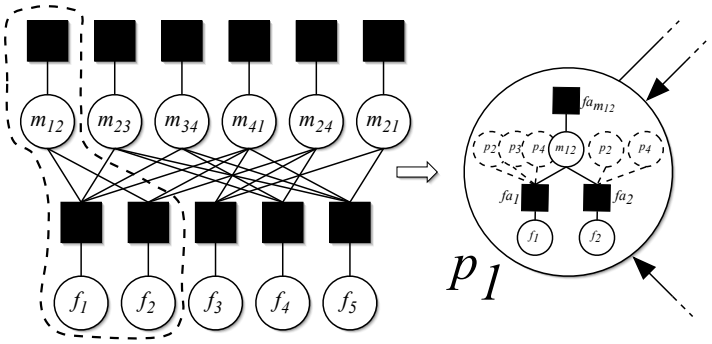

Figure 8. Creating a local factor-graph in the PDMS (here for peer p_1)

values for $f^i_{\circlearrowleft}$ and $f^j_{\rightrightarrows}$). Hence, each peer p only needs to store a fraction of the global factor-graph, fraction selected as follows:

```
for all outgoing mapping m
   add m.factor to local factor-graph;
   add m.variable to m.factor;
   for all feedbackMessage f containing m
      add f.factor to m.variable;
      if f.isPositive
         add f.variable(+) to f.factor;
      else if feedback.isNegative
         add f.variable(-) to f.factor;
      for all mapping m' in feedback except m
         add virtual peer m'.peer to f.factor;
```

where $feedbackMessage$ stands for all feedback messages received from neighboring peers (resulting from probes flooded within a certain TTL throughout the neighborhood or from analyzing standard forwarded queries). Figure 8 shows how p_1 from Figure 7 would store its local factor-graph.

Note that, depending on the PDMS, one can choose between two levels of granularity for storing factor-graphs and computing related probabilistic value: coarse granularity – where peers only store one factor-graph per mapping and where they derive only one global value on the correctness of the mapping – and fine granularity – where peers store one instance of the local factor-graph per attribute in the mapping, and where they derive one probabilistic quality value per attribute. We suppose we are in the latter situation but show derivations for one attribute only. Values for other attributes can be derived in a similar fashion.

5.2. On Cycles in PDMS Factor-Graphs

Cycles appear in PDMS factor-graphs as soon as two mappings belong to two identical cycles or parallel paths in the PDMS. See for example the PDMS in Figure 6, where m_{12}

and m_{41} both appear in cycles $p_1 - p_2 - p_4 - p_1$ and $p_1 - p_2 - p_3 - p_4 - p_1$, hence creating a cycle $m_{12} - factor(f_1) - m_{41} - factor(f_2) - m_{12}$ in the factor-graph. As mentioned above, the results of the sum-product algorithm operating in a factor-graph with cycles cannot (in general) be interpreted as exact function summaries.

One well-known approach to circumvent this problem is to transform the factor-graph by regrouping nodes (*clustering* or *stretching* transformations) to produce a factor tree. In our case, this would result in regrouping all mappings having more than one cycle or parallel path in common; this is obviously inapplicable in practice, as this would imply introducing central components in the PDMS to regroup (potentially large) sets of independent peers. Instead, we rely on iterative, decentralized message passing schedules (see below) to estimate marginal functions in a concurrent and efficient way. We show in Section 6 that those evaluations are sufficiently accurate to make sensible decisions on the mappings in practice.

5.3. Embedded Message Passing Schedules

Given its local factor-graph and messages received from its neighborhood, a peer can locally update its belief on the mappings by reformulating the sum-product algorithm (Section 4.1) as follows:

local message from factor fa_j to mapping variable m_i:

$$\mu_{fa_j \to m_i}(m_i) = \sum_{\sim\{m_i\}} \left(fa_j(X) \prod_{p_k \in n(fa_j)} \mu_{p_k \to fa_j}(p_k) \prod_{m_l \in n(fa_j)\setminus\{m_i\}} \mu_{m_l \to fa_j}(m_l) \right)$$

local message from mapping m_i to factor $fa_j \in n(m_i)$:

$$\mu_{m_i \to fa_j}(m_i) = \prod_{fa \in n(m_i)\setminus\{fa_j\}} \mu_{fa \to m_i}(m_i)$$

remote message for factor fa_k from peer p_0 to peer $p_j \in n(fa_k)$:

$$\mu_{p_0 \to fa_k}(m_i) = \prod_{fa \in n(m_i)\setminus\{fa_k\}} \mu_{fa \to m_i}(m_i)$$

Posterior correctness of local mapping m_i:

$$P(m_i|\{\mathcal{F}\}) = \alpha \left(\prod_{fa \in n(m_i)} \mu_{fa \to m_i}(m_i)) \right)$$

where $alpha$ is a normalizing factor ensuring that the probabilities of all events sum to one (i.e., making sure that $P(m_i = correct) + P(m_i = incorrect) = 1$) .

In cycle-free PDMS factor-graphs (i.e., trees), exact messages can be propagated from mapping variables to the rest of the network in at most two iterations (due to the specific topology of our factor-graph). Thus, all inference results will be exact in two iterations.

For the more general case of PDMS factor-graph with cycles, we are stuck at the beginning of computation since every peer has to wait for messages from other peers.

We resolve this problem in a standard manner by considering that all peers virtually received a unit message (i.e., a message representing the unit function) from all other peers appearing in their local factor-graphs prior to starting the algorithm. From there on, peers derive probabilities on the correctness of their local mappings and send messages to other peers as described above. We show in Section 6 that for PDMS factor-graphs with cycles, the algorithm converges to very good approximations of the exact values obtained by a standard global inference process. Peers can decide to send messages according to different schedules depending on the PDMS; we detail below two possible schedules with quite different performance in terms of communication overhead and convergence speed.

5.3.1. Periodic Message Passing Schedule

In highly dynamic environments where databases, schemas and schema mappings are constantly evolving, appearing or disappearing, peers might wish to act proactively in order to get results on the correctness of their mappings in a timely fashion. In a Periodic Message Passing Schedule, peers send remote messages to all peers p_i appearing in their local factor-graph every time period τ. This corresponds to a new round of the iterative sum-product algorithm. This periodic schedule induces some communication overhead (a maximum of $\sum_{c_i}(l_{c_i}-1)$ messages per peer every τ, where c_i represent all mapping cycles passing through the peer and l_{c_i} the length of the cycles) but guarantees our methods to converge within a given time-frame dependent on the topology of the network (see also Section 6). Note that τ should be chosen according to the network churn in order to guarantee convergence in highly dynamic networks. Its exact value may range from a couple of seconds to weeks or months depending on the exact situation.

5.3.2. Lazy Message Passing Schedule

A very nice property of the iterative message passing algorithm is that it is tolerant to delayed or lost messages. Hence, we do not actually require any kind of synchronization for the message passing schedule; Peers can decide to send a remote message whenever they want without endangering the global convergence of the algorithm (the algorithm will still converge to the same point, simply slower, see next section). We may thus take advantage of this property to totally eliminate any communication overhead (i.e., number of additional messages sent) induced by our method by piggybacking on query messages. The idea is as follows: every time a query message is sent from one peer to another through a mapping link m_i, we append to this query message all messages $\mu(m_i)$ pertaining to the mapping being used. In this case, the convergence speed or our algorithm is proportional to the query load of the system. This may be the ideal schedule for query-intensive or relatively static systems.

5.4. Prior Belief Updates

Our computations always take into account the mapping factors (top layer of a PDMS factor-graph). These factors represent any local, prior knowledge the peers might possess on their mappings. For example, if the mappings were carefully checked and validated by a domain expert, the peer might want to set all prior probabilities on the correctness of the mappings to one to ensure that these mappings will always be treated as correct.

In most cases, however, the peers only have a vague idea (e.g., presupposed quality of the alignment technique used to create the mappings) on the priors related to their surrounding mappings initially. As the network of mappings evolves and time passes, however, the peers start to accumulate various posterior probabilities on the correctness of their mappings thanks to the iterative message passing techniques described above. Actually, the peers get new posterior probabilities on the correctness of the mappings as long as the network of mappings continues to evolve (e.g., as mappings get created, modified or deleted). Thus, peers can decide to modify their prior belief by taking into account the evidences accumulated in order to get more accurate results in the future. This corresponds to learning parameters in a probabilistic graphical model when some of the observations are missing. Several techniques might be applied to this type of problem (e.g., Monte Carlo methods, Gaussian approximations). We propose in the following a simple Expectation-Maximization [20] process which looks as follows:

- Initialize the prior probability on the correctness of the mapping taking into account any prior information on the mapping. If no information is available for a given mapping, start with $P(m = correct) = P(m = incorrect) = 0.5$ (maximum entropy principle).
- Gather posterior evidences $P_k(m = correct|\{\mathcal{F}_k\})$ on the correctness of the mapping thanks to cycle analyses and message passing techniques. Treat these evidences as new observations for every change of the local factor-graphs (i.e., new feedback information, new, modified or lost cycle or parallel path)
- After each change of the local factor-graph, update the prior belief on the correctness of the mapping m given previous evidences $P_k(m = correct|\{\mathcal{F}_k\})$ in the following way:

$$P(m = correct) = \sum_{i=1}^{k} P_i(m = correct|\{\mathcal{F}_i\})k^{-1}$$

Hence, we can make the prior values slowly converge to a local maximum likelihood to reflect the fact that more and more evidences are being gathered about the mappings as the mapping network evolves.

5.5. Introductory Example Revisited

Let us now come back to our introductory example and describe in more detail what happened. Imagine that the network of databases was just created and that the peers have no prior information on their mappings. By sending probe queries with $TTL \geq 4$ through its two mapping links, p_2 detects two cycles and one parallel path, and gets all related feedback information. For the attribute *Creator*:

$$\boldsymbol{f}_{\circlearrowleft}^{\mathbf{1}+} : m_{12} \rightarrow m_{23} \rightarrow m_{34} \rightarrow m_{41}$$
$$\boldsymbol{f}_{\circlearrowleft}^{\mathbf{2}-} : m_{12} \rightarrow m_{24} \rightarrow m_{41}$$
$$\boldsymbol{f}_{\rightrightarrows}^{\mathbf{3}-} : m_{24} \| m_{23} \rightarrow m_{34}$$

p_2 constructs a local factor-graph based on this information and starts sending remote messages and calculating posterior probabilities on its mappings according to the

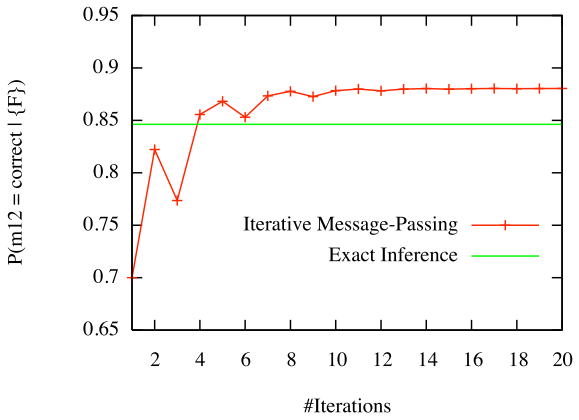

Figure 9. Convergence of the iterative message passing algorithm compared to exact inference (example graph, priors at 0.7, $\Delta = 0.1$, f_1^+, f_2^-, f_3^-)

schedule in place in the PDMS. Δ, the probability that two or more mapping errors get compensated along a cycle, is here approximated to 1/10: if we consider that the schema of p_2 contains eleven attributes, and that mapping errors map to a randomly chosen attribute (but obviously not the correct one), the probability of the last mapping error compensating any previous error is 1/10, thus explaining our choice. After a handful of iterations, the posterior probabilities on the correctness of p_2's mappings towards p_3 and p_4 converge to 0.59 and 0.3 respectively. The second mapping has been successfully detected as faulty for the given attribute, and will thus not be used to forward query q_1 ($\theta_i = 0.5$). The query will however reach all other databases by being correctly forwarded through $p_2 \rightarrow p_3$, $p_3 \rightarrow p_4$ and finally $p_4 \rightarrow p_1$. As the PDMS network evolves, p_2 will update its prior probabilities on the mapping toward p_3 and p_4 to 0.55 and 0.4 respectively to reflect the knowledge gathered on the mappings so far.

6. Performance Evaluation

We present below series of results related to the performance of our approach. We start by giving a couple of results pertaining to simple PDMS networks before analyzing larger sets of automatically generated networks.

6.1. Performance Evaluation on the Example Graph

6.1.1. Convergence

As previously mentioned, our inference method is exact for cycle-free PDMS factor-graphs. For PDMS factor-graphs with cycles, our embedded message passing scheme converges to approximate results in ten iterations usually. Figure 9 illustrates a typical convergence process for the example PDMS factor-graph of Figure 6 for schemas of about ten attributes (i.e., Δ set to 0.1), prior beliefs at 0.7 and cycle feedback as follows: f_1^+, f_2^-, f_3^-.

6.1.2. Fault-Tolerance

As mentioned earlier for the lazy message passing schedule, our scheme does not requires peers to be synchronized to send their messages. To simulate this property, we randomly discard messages during the iterative message passing schedule and observe the resulting effects. Figure 10 shows the results if we consider, for every message to be sent, a probability $P(send)$ to send it only (example network, $\Delta = 0.1$, priors at 0.8, f_1^+, f_2^-, f_3^-). We observe that our method always converges, even for cases where 90% of the messages get discarded, and that the number of iterations required in order for our algorithm to converge grows linearly with the rate of discarded messages.

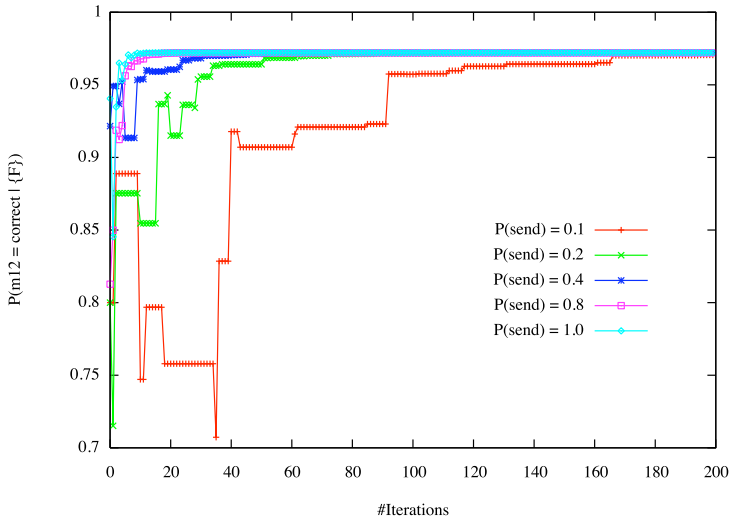

Figure 10. Robustness against faulty links, with probabilities of correctly sending a message ranging from 10% to 100% (example graph, priors at 0.8, $\Delta = 0.1$, f_1^+, f_2^-, f_3^-)

6.2. Performance Evaluation on Random PDMS Networks

To test our heuristics on larger networks, we create schema nodes and add edges by randomly choosing a distinct pair of nodes for each undirected mapping we wish to include. We obtain irreflexive, non redundant and undirected Poisson-distributed graphs in this manner. We randomly pick a certain proportion of mappings and create erroneous links. Also, we randomly select a given percentage of cycle feedback for which two or more errors get compensated. Finally, we run our iterative message passing heuristics on the resulting graphs and determine for each mapping whether it is correct or not (most probable value of $P(m_i = correct|\{\mathcal{F}\})$. The results are given in terms of *precision* values, where precision is defined as the ratio of the number of correctly evaluated mappings over the total number of mappings evaluated. Each result is given as an average value calculated over twenty consecutive rounds, with a confidence interval corresponding to a confidence level of 95%.

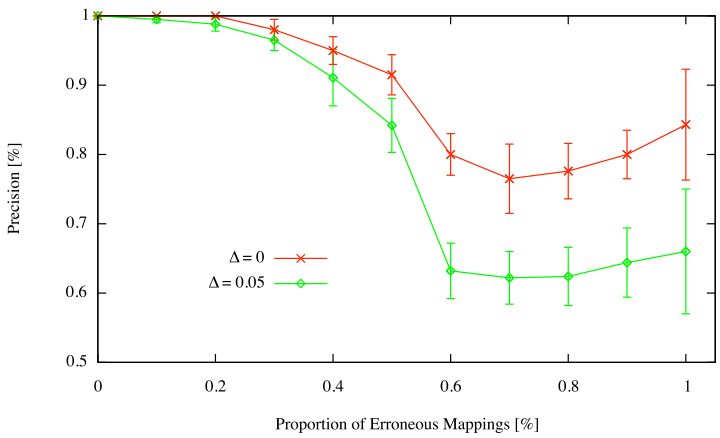

Figure 11. Precision of erroneous mapping detection on random networks of 50 schemas and 200 mappings, with a varying proportion of erroneous mappings, two values of Δ, TTL = 5 to detect cycles and without any a priori information

6.2.1. Performance with an Increasing Proportion of Erroneous Mappings

Figure 11 provides results corresponding to networks of 50 schemas and 200 mappings, with an increasing percentage of erroneous mappings and for relatively small schemas ($\Delta = 5\%$). Our methods work surprisingly well for low densities of incorrect mappings, with 98% or more of correct decisions for networks with less than 30% of erroneous mappings. For networks with a larger proportions of incorrect mappings, the results are less spectacular but still satisfying with precision values above 60%. Note that these values are obtained automatically, via a totally decentralized process and without any prior information on the mappings. Compensating errors make it difficult to detect all errors in networks with many erroneous mappings (cycles which should be treated as negative are in fact seen as positive). This fact is highlighted by a second curve in Figure 11 ($\Delta = 0$), corresponding to very large schemas, where compensating errors can be neglected and where it is much easier to make sensible decisions on networks with very high proportions of incorrect mappings.

6.2.2. Precision with an Increasing Number of Mappings

Figure 12 provides results corresponding to networks of 50 schemas and an increasing number of mappings between the schemas. For sparse networks (e.g., 50 mapping links, corresponding to one mapping per schema on average), few cycles can be detected and thus little feedback information is available. As more and more mappings are created, more feedback information gets available thus making it easier to take sensible decisions on the correctness of the mappings. Dense networks have a very high number of long cycles (e.g., in scale-free networks, where the number of large loops grows exponentially with the size of the loops considered [18]); the longer the cycle, however, the less interesting it is from an inference point of view as it is related to a higher number of mapping variables (and hence represents less precise information). Thus, peers should always be cautious to analyze the most pertinent feedback information only, pertaining to cycles or parallel paths as small as possible, and to keep their TTL for detecting cycles and parallel

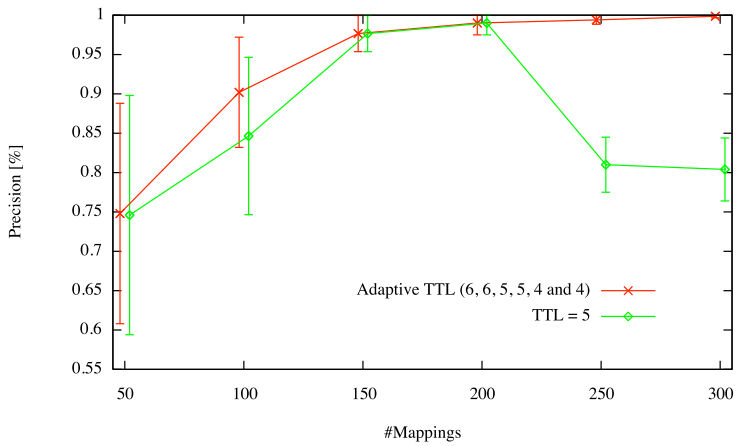

Figure 12. Precision of erroneous mapping detection on random networks of 50 schemas and a varying number of mappings, with a proportion of 20% of erroneous mappings, $\Delta = 0.05$, without any a priori information but with different TTL values for detecting the cycles

paths relatively small; to highlight this fact, Figure 12 shows two curves: one for a fixed TTL of 5 and one with an adaptive TTL (6 for 50 to 100 mappings, 5 for 150 to 200 mappings and 4 from 250 mappings). Adapting the TTL value is important in two situations: first, in sparse networks where peers should try to detect longer cycles in order to get more feedback information (e.g. for 100 mappings in Figure 12, where a TTL of 6 leads to better results than a TTL of 5). In very sparse networks, however, there are simply too few mappings to detect a sufficient number of cycles, even for large TTL values (e.g., for 50 mappings in Figure 12). Second, in dense networks, where precious information given by short cycles can rapidly be diluted by taking into account longer cycles (e.g., for 300 mappings in Figure 12, where more than 20000 cycles of length 5 can be discovered, leading to poorer results if all taken into account). From a local perspective, peers should thus start with low TTL values and increase their TTL only when very few cycles are discovered. This also ensures the scalability of our approach: peers can concentrate on their direct vicinity and do not need to analyze the network as a whole.

7. Conclusions and Future Work

As distributed database systems move from static, controlled environments to highly dynamic, decentralized settings, we are convinced that correctly handling uncertainty and erroneous information will become a key challenge for improving the overall quality of query answering schemes. The vast majority of approaches are today centered around global and deductive methods, which seem quite inappropriate to maximize the performance of systems that operate without any form of central coordination. Contrary to these approaches, we consider an abductive, non-monotonic reasoning scheme, which reacts to observations and inconsistencies by propagating belief in a decentralized way. Our approach is computationally efficient as it is solely based on sum-products operations. Also, we have shown its high effectiveness by evaluating it on sets of randomly

generated database networks. We are currently implementing our approach in our Semantic Overlay Network called GridVine [14] and plan to analyze the computational overhead and scalability properties of our iterative message passing approach in dynamic environments. Furthermore, we are currently interested in testing other inference techniques (e.g., generalized belief propagation [21], or techniques constructing a junction tree in a distributed way [22]) in order to determine the most efficient way of performing inference in our decentralized database setting.

References

[1] K. Aberer and P. Cudré-Mauroux. Semantic Overlay Networks. In *International Conference on Very Large Data Bases (VLDB)*, 2005.

[2] K. Aberer (ed.). Special issue on peer to peer data management. *ACM SIGMOD Record*, 32(3), 2003.

[3] P. Bouquet et al. Specification of a common framework for characterizing alignment. In *KnowledgeWeb Deliverable 2.2.1*, *http://knowledgeweb.semanticweb.org*.

[4] D. L. McGuinness and F. van Harmelen (eds). Owl web ontology language overview. W3C Recommendation, 2004.

[5] J. Euzenat et al. State of the art on current alignment techniques. In *KnowledgeWeb Deliverable 2.2.3*, *http://knowledgeweb.semanticweb.org*.

[6] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. Start making sense: The Chatty Web approach for global semantic agreements. *Journal of Web Semantics*, 1(1), 2003.

[7] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *International World Wide Web Conference (WWW)*, 2003.

[8] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Workshop on the Web and Databases (WebDB)*, 2002.

[9] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *International World Wide Web Conference (WWW)*, 2002.

[10] I. Tatarinov and A. Halevy. Efficient Query Reformulation in Peer-Data Management Systems. In *SIGMOD Conference*, 2004.

[11] I. Tatarinov, Z. Ives, J. Madhavan amd A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyaska, G. Miklau, and P. Mork. The Piazza Peer Data Management Project. *ACM SIGMOD Record*, 32(3), 2003.

[12] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record*, 32(3), 2003.

[13] B. C. Ooi, Y. Shu, and K.-L. Tan. Relational Data Sharing in Peer-based Data Management Systems. *ACM SIGMOD Record*, 32(3), 2003.

[14] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. van Pelt. Gridvine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference*, 2004.

[15] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 2001.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[17] K. M. Murphy, Y. Weiss, and M. I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence (UAI)*, 1999.

[18] G. Bianconi and M. Marsili. Loops of any size and hamilton cycles in random scale-free networks. In *cond-mat/0502552 v2*, 2005.

[19] P. Cudré-Mauroux, K. Aberer, and Andras Feher. Probabilistic message passing in peer data managemen systems. In *International Conference on Data Engineering (ICDE)*, 2006.
[20] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39, 1977.
[21] J.S. Yedidia, W.T. Freeman, and Y Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems (NIPS)*, 13, 2000.
[22] M.A. Paskin and C.E. Guestrin. A robust architecture for distributed inference in sensor networks. In *Intel Research Technical Report IRB-TR-03-039*, 2004.