Probabilistic Message Passing in Peer Data Management Systems[*]

Philippe Cudré-Mauroux
School of Computer and Communication Sciences
EPFL – Switzerland

Karl Aberer
School of Computer and Communication Sciences
EPFL – Switzerland

Andras Feher
Fachbereich Informatik
T.U. Darmstadt
Germany

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.

PDMS approaches proposed so far make the implicit assumption that all mappings used in this way are correct. This obviously cannot be taken as granted in typical PDMS settings where mappings can be created (semi) automatically by independent parties. In this work, we propose a totally decentralized, efficient message passing scheme to automatically detect erroneous mappings in 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 a preliminary evaluation of our techniques on sets of both automatically-generated and real-world schemas.

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-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 paper, we focus on the Peer Data Management Systems (PDMS) approach, which recently drew considerable attention in this context [4].

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

[*]The work presented in this paper 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 and was (partly) 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 European project Evergrow No 001935.

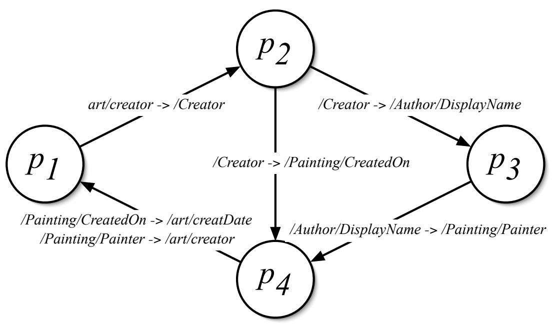
control on schema designs. As a result, we can expect irreconcilable differences on conceptualizations (e.g., epistemic or metaphysical differences on a given modelization problem, see also [18]) among the databases. Also, the limited expressivity of the mappings, usually defined as queries or using an ontology definition language like OWL, precludes the creation of correct mappings in many situation (e.g., mapping an attribute onto a relation). Finally, given the vibrant activity on (semi-) automatic alignment techniques (see [11] 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.

1.1 Our Contribution

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 a preliminary evaluation of our technique applied on both generated and real-world data.

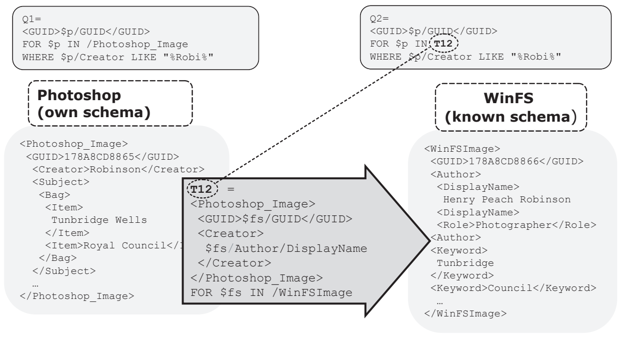
1.2 An Introductory Example

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 1. 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 2 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.



p_1
p_2
p_3
p_4
art/creator -> /Creator
/Creator -> /Author/DisplayName
/Creator -> /Painting/CreatedOn
/Painting/CreatedOn -> /art/creatDate
/Painting/Painter -> /art/creator
/Author/DisplayName -> /Painting/Painter

Figure 1. A simple directed PDMS network of four peers and five schema mappings, here depicted for the attribute *Creator*



```
Q1=
<GUID>$p/GUID</GUID>
FOR $p IN /Photoshop_Image
WHERE $p/Creator LIKE "%Robi%"
```

```
Q2=
<GUID>$p/GUID</GUID>
FOR $p IN T12
WHERE $p/Creator LIKE "%Robi%"
```

```
<Photoshop_Image>
 <GUID>178A8CD8865</GUID>
  <Creator>Robinson</Creator>
  <Subject>
   <Bag>
    <Item>
     Tunbridge Wells
    </Item>
    <Item>Royal Council</I
   </Bag>
  </Subject>
  …
</Photoshop_Image>
```

```
T12 =
<Photoshop_Image>
 <GUID>$fs/GUID</GUID>
 <Creator>
  $fs/Author/DisplayName
 </Creator>
</Photoshop_Image>
FOR $fs IN /WinFSImage
```

```
<WinFSImage>
 <GUID>178A8CD8866</GUID>
 <Author>
  <DisplayName>
   Henry Peach Robinson
  <DisplayName>
  <Role>Photographer</Role>
 <Author>
 <Keyword>
  Tunbridge
 </Keyword>
 <Keyword>Council</Keyword>
  …
</WinFSImage>
```

Figure 2. An example of schema mapping (here between peers p_2 and p_3) expressed as an XQuery

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.

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 1). Luckily for our user, the PDMS system he is using implements our message passing 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.

2 Problem Definition

We model PDMS as collections of peers; Each individual peer p represents a database storing data according to a distinct structured schema. Note that a peer could in practice represent a (potentially large) cluster of databases all adhering to the same 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., they 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 [3]) 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 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.

3.1 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 [12]. 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 3, 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.,

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

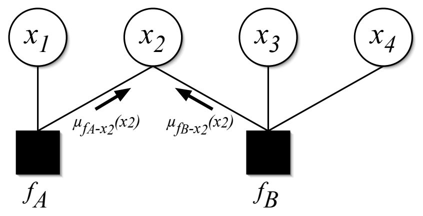


Figure 3. A simple factor-graph of four variables and two factors

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

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 3). 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:

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

local factor f to variable x

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

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 [15]. 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. Finally, note that Belief Propagation as introduced by Judea Pearl [20] is actually a specialized case of a standard message passing sum-product algorithm.

3.2 On Factor-Graphs in Undirected PDMS

In the following, we explain how one can model a network of mapping as a factor-graph. These factor-graphs will in turn be used in Section 4 to derive quality measures for the various mappings in the network.

3.2.1 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 [7].

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 [3]). In the current case, we consider that the probability on the correctness of a mapping link 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 to be taken 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 4 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 4). The peers are interconnected

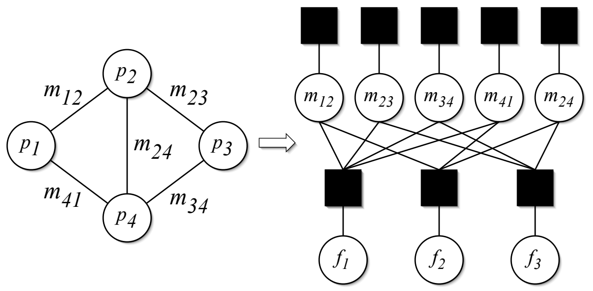


Figure 4. Modeling an undirected network of mappings as a factor-graph

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:

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

The right-hand side of Figure 4 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 4, where all three feedbacks are correlated).

3.3 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. Factors from directed mapping cycles $m_0 \rightarrow m_1 \rightarrow, \ldots \rightarrow m_{n-1}$ are defined in exactly the same ways as explained above for the undirected case.

Parallel mapping paths occur when two different series of mappings m' and m'' share the same source and destination, e.g., $m'_0 \rightarrow m'_1 \rightarrow \ldots \rightarrow m'_{n'-1}$ and $m''_0 \rightarrow m''_1 \rightarrow \ldots \rightarrow m''_{n''-1}$, with m'_0 and m''_0 departing from the same peer and $m'_{n'-1}$ and $m''_{n''-1}$ arriving at the same peer. Those two parallel paths would be considered as forming an undirected cycle in an undirected network, but cannot be considered as such here due to the restriction on the direction of the mapping operations in a network of directed mappings.

If a query q is forwarded through both parallel paths, the destination peer p_{n-1} can compare both $q' =$

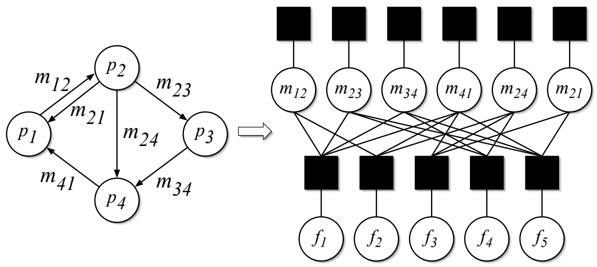


Figure 5. Modeling a directed network of mappings as a factor-graph

$m'_{n'-1}(\ldots(m'_0(q)))$ and $q'' = m''_{n''-1}(\ldots(m''_0(q)))$. As for the undirected cycle presented above, three cases can occur when comparing an operations $op(a'_i)$ of q' to its corresponding operation $op(a''_j)$ in q'': positive feedback ($a'_i \equiv a''_j$), negative feedback ($a'_i \neq a''_j$) of neutral feedback ($a'_i = \perp$ and/or $a''_j = \perp$).

The conditional probability function for receiving positive feedback $f^{+}_{\rightrightarrows}$ through parallel paths knowing the correctness of the sets of mappings $\{m'\} = \{m'_0, \ldots, m'_{n'-1}\}$ and $\{m''\} = \{m''_0, \ldots, m''_{n''-1}\}$ is:

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

Figure 5 below shows an example of 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:

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

As for the undirected case, the right-hand side of Figure 5 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.

4 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 3.2 and 3.3, 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.

4.1 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 2). 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 values for $f^{i}_{\circlearrowleft}$ and $f^{j}_{\rightrightarrows}$, see preceding section). 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;
```

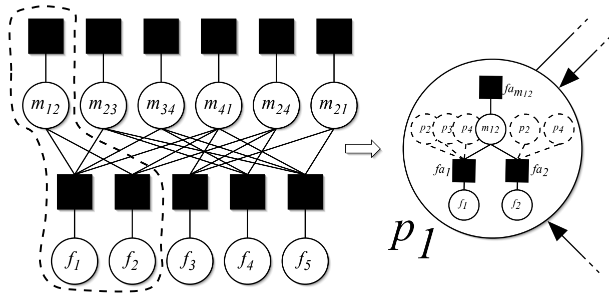


Figure 6. Creating a local factor-graph in the PDMS (here for peer p_1)

where $feedbackMessage$ stands for all cycle of parallel path feedback messages received from neighboring peers (resulting from probes flooded within a certain TTL throughout the neighborhood or from analyzing standard forwarded queries). Figure 6 shows how p_1 from Figure 5 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 only one attribute in the following. Values for other attributes can be derived in a similar fashion.

4.2 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 4, 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 (see also Section 7 for a discussion on this topic). 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 5 that those evaluations are sufficiently accurate to make sensible decisions on the mappings in practice.

4.3 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 3.1) as follows:

local message from factor fa_j to mapping variable m_i:

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

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:

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

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 5 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.

4.3.1 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 5). 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.

4.3.2 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 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.

4.4 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 [8] 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.

4.5 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 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 here 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.

5 Performance Evaluation

We present below series of results pertaining to the performance of our approach. We proceed in two phases: We first report on sets of simulations to highlight some of the specificities of our approach before presenting results obtained on a set of real-world schemas.

5.1 Performance Analyses

5.1.1 Convergence

As previously mentioned, our approach 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 7 below illustrates a typical convergence process for the example PDMS factor-graph of Figure 4 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^-.

Note that the accuracy does not suffer from longer cycles: Figure 9 shows the relative error between our iterative and decentralized scheme and a global inference process. The relative error is computed for our example graph ($\Delta = 0.1$, priors at 0.8, f_1^+, f_2^-, f_3^-, 10 iterations), and

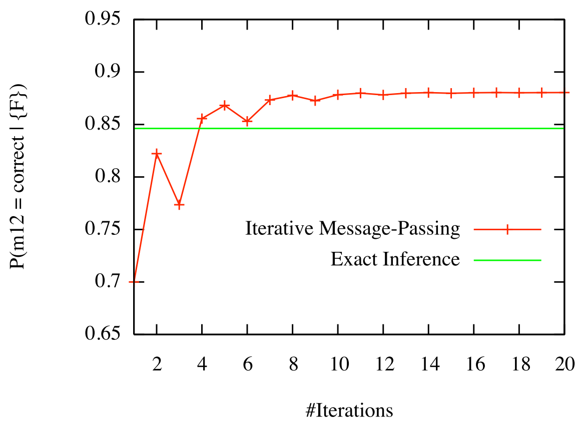


Figure 7. Convergence of iterative message passing algorithm (example graph, priors at 0.7)

by successively adding a new peer as depicted in Figure 8. The relative error is bigger for very short cycles but never reaches 6% (these results are quite typical of iterative message passing schemes).

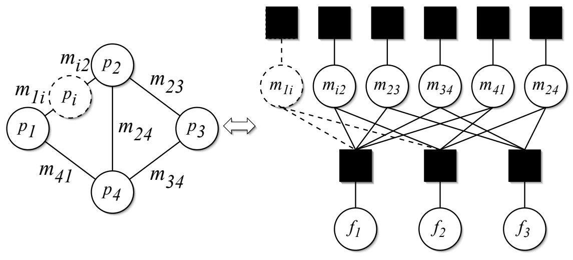


Figure 8. Adding nodes iteratively to increase the cycle length

5.1.2 Cycle Length Impact

As cycles in the PDMS get bigger, so do the number of variables appearing in the feedback factors. Thus, shorter cycles provide more precise evidences on the correctness (or incorrectness) of a mapping than longer cycles due to the inherent uncertainty pertaining to each mapping variable. Figure 10 demonstrates this claim for simple cyclic PDMS networks of two to twenty nodes (priors at 0.5, positive feedback, 2 iterations [cycle-free factor-graph]). The results are quite naturally influenced by the value of Δ, but in the end, cycles greater than ten mappings always end up by providing very little evidence on the correctness of the

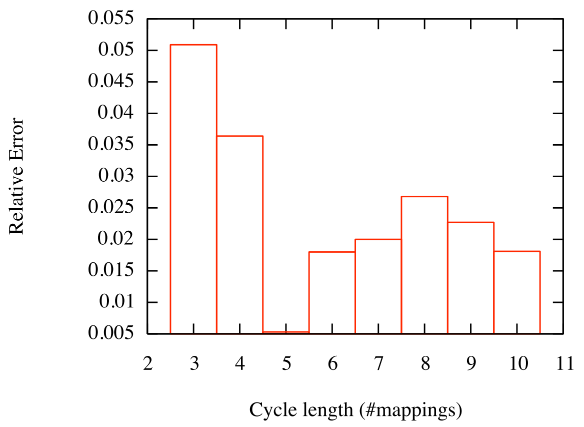


Figure 9. Relative error of iterative message passing algorithm for various cycle lengths (10 iterations, example graph, priors at 0.8)

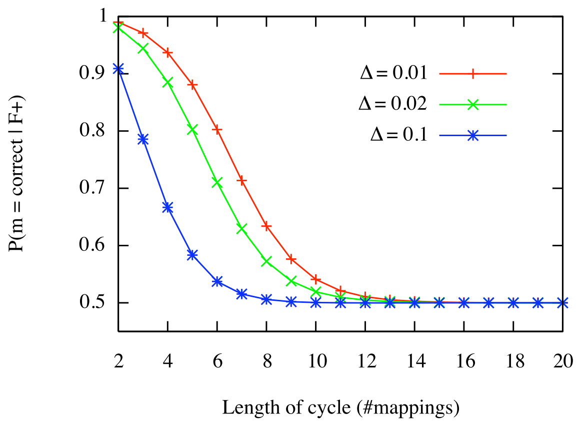


Figure 10. Impact of the cycle length on the posterior probability, here for a simple positive cycle graph of a varying number of links for three values of Δ

mappings, even for bigger schemas for which compensating errors are infrequent (e.g., Figure 10 for Δ=0.01).

Scale-free networks are known to have a number of large loops growing exponentially with the size of the loops considered [7]. Generally speaking, this would imply exponentially growing PDMS factor-graphs as well for large-scale networks. However, as we have just seen, the impact of cycles on the posterior probabilities diminishes rapidly with the size of the cycles.

Peers can locally make a sensible decision on the tradeoff between the maximal length of the cycles they consider and the precision of the results they can get as follows: They start by considering the direct vicinity of their mappings only and send probes with low TTL values. As they gradually increase the TTL of their probes to detect longer cycles, they monitor the impact of the new cycles they discover on the posteriors of their mappings. Peers can stop this expansion process as soon as changes induced by the new cycles get insignificant. At this stage, peers are guaranteed to have discovered the most pertinent cycles (or parallel paths) for their mappings anyway. In practice, the threshold on the cycle length might vary, but it always remains low (i.e., five to ten) for dense graphs, even in very large-scale environments.

5.1.3 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 11 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.

5.2 Applying Message Passing on Real-World Schemas

We have developed a tool to test our methods on real-world schemas and mappings. This tool can import OWL schemas (serialized in RDF/XML) and simple RDF mappings (following the format introduced in [18]). It can also create new mappings using the simple alignment techniques described in [10]. Once schemas and alignments have been defined, the tool creates peers, automatically detects cycles, transforms the setting into a PDMS factor-graph and starts sending messages in order to get posterior quality values on the mappings as described above.

We report on preliminary experiments based on a set of standard schemas from the EON Ontology Alignment Contest (http://co4.inrialpes.fr/align/Contest/). The setting is as follows: we first import various ontologies related to bibliographic data: the so-called reference ontology (101), a similar ontology but translated in French (221), the Bibtex ontology from M.I.T., the Bibtex ontology from UMBC, and two other bibliographic ontologies from INRIA and Karlsruhe. Each of these ontologies is composed of about thirty concepts (attributes). We then start our automatic alignment techniques to create mappings between this set of schemas. We launch the iterative message passing algorithm on the

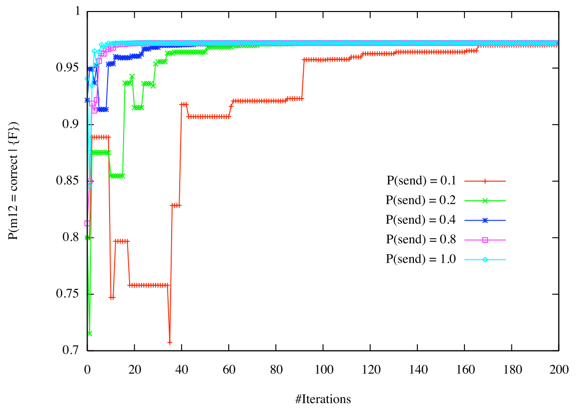


Figure 11. Robustness against faulty links (lost messages)

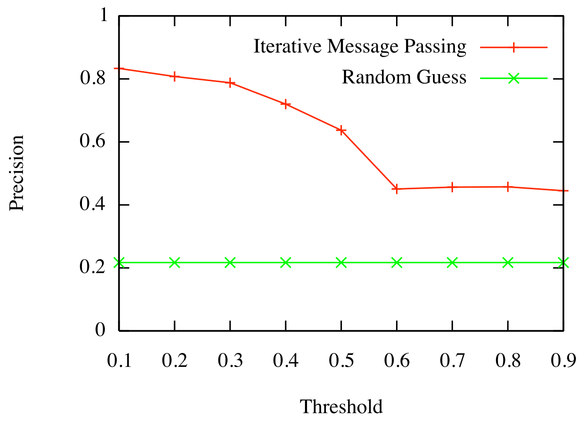


Figure 12. Precision results of the Message Passing approach with a varying threshold θ

resulting PDMS factor-graph and try to detect erroneous mappings automatically. Finally, we ask a human expert to manually judge on the quality of the results.

Figure 12 shows the precision of our approach for 396 generated mappings w.r.t. the threshold θ used to determine whether a mapping is correct or not (priors at 0.5, Δ=0.1). Here, the precision is defined as the number of mappings correctly detected as being erroneous over the total number of mappings detected as erroneous. The results are particularly appealing for low θ values, where our methods predicts erroneous mappings with 80% or more accuracy. As θ grows, however, more and more mixed or uncertain results are being considered, thus quite naturally lowering the precision. We observe a phase-transition at $\theta = 0.6$. At this point, 50% of the 86 erroneous mappings have been automatically discovered by the algorithm and increasing θ beyond this will not lead to significantly better results. Still, even for high threshold values, our method is significantly better than random guesses.

These preliminary results are quite encouraging, considering the facts that no prior information was provided on the mappings, and that only one complete round of the algorithm could be completed on this static network (i.e., no update on prior beliefs).

6 Related Work

P2P data management techniques recently drew considerable attention (see [1] or [4] for an introduction). In [6], Bernstein et al. formulated requirements for P2P databases in terms of local mappings and semi-automatic solutions for data integration. Edutella [16], based on a super-peer topology, was one of the early systems deployed. Piazza [21, 22] is a PDMS system which provides efficient query reformulations algorithms for relational and semi-structured data models. The Hyperion [5] project relies on rules to propagate data in a PDMS network according to mapping tables. PeerDB [17] examines the problem of sharing relational data in P2P networks from an information retrieval perspective. None of these works directly addresses the problem of finding or handling faulty mapping links.

We can draw a parallel between our method and global schema alignment methods (see [11] for an overview). Similarity Flooding [14] is a graph matching technique based on a fixpoint-computation. The Glue system [9] learns classifiers for classes from instances in order to evaluate the joint probability distributions of the instances. Corpus-based Schema Matching [13] relies on statistics about elements and their relationships to infer constraints on the schema mappings. None of these approaches takes advantages of inconsistencies detected in the mapping network nor do they use inference probabilistic networks.

Our approach is quite naturally based on some of our previous ideas [2, 3] on analyzing the network of mappings. Our previous work, however, was not concerned with parallel paths, prior beliefs, efficient message passing schemes, belief propagation or probabilistic networks; it was computationally much more expensive and, above all, ignored all interdependencies among the mappings and cycles, thus resulting in comparatively poorer results. As an example, applying our former heuristics to the introductory example graph would result in disqualifying all three mappings on the left (while only one is erroneous) while, by considering all correlations between the mappings and cycles, our current approach yields to much better results (correct inference for all five mappings).

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 local 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 or inconsistencies by globally propagating belief in a decentralized way. Our approach is computationally efficient as it is solely based on sum-products operations. Also, we have proven its usefulness by evaluating its performance on a set of real-world schemas.

We are currently testing our heuristics on larger automatically-generated PDMS settings. We are particularly interested in understanding the relation between exact inference and our approximate results in those environments. We are also analyzing the computational overhead and scalability properties of other inference techniques (e.g., generalized belief propagation [23], or techniques constructing a junction tree in a distributed way [19]). Finally, as global interoperability is always challenged by the dynamics of the mapping network, we plan to analyze the tradeoff between the efforts required to maintain the probabilistic network in a coherent state and the potential gain in terms of relevance of results.

References

[1] K. Aberer and P. Cudré-Mauroux. Semantic Overlay Networks. In *International Conference on Very Large Data Bases (VLDB)*, 2004.

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39, 1977.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *International World Wide Web Conference (WWW)*, 2002.

[10] J. Euzenat. An api for ontology alignment. In *International Semantic Web Conference (ISWC)*, 2004.

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

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

[13] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *International Conference on Data Engineering (ICDE)*, 2005.

[14] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *International Conference on Data Engineering (ICDE)*, 2002.

[15] 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.

[16] 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.

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

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

[19] M. Paskin and C. Guestrin. A robust architecture for distributed inference in sensor networks. In *Intel Research Technical Report IRB-TR-03-039*, 2004.

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

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

[22] I. Tatarinov, Z. Ives, J. M. 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.

[23] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems (NIPS)*, 13, 2000.