idMesh: Graph-Based Disambiguation of Linked Data

Philippe Cudré-Mauroux
CSAIL
MIT – USA
pcm@csail.mit.edu

Parisa Haghani
I&C
EPFL – Switzerland
parisa.haghani@epfl.ch

Michael Jost
I&C
EPFL – Switzerland
michael.jost@epfl.ch

Karl Aberer
I&C
EPFL – Switzerland
karl.aberer@epfl.ch

Hermann de Meer
U. Passau – Germany
demeer@fmi.uni-passau.de

ABSTRACT

We tackle the problem of disambiguating entities on the Web. We propose a user-driven scheme where graphs of entities – represented by globally identifiable declarative artifacts – self-organize in a dynamic and probabilistic manner. Our solution has the following two desirable properties: i) it lets end-users freely define associations between arbitrary entities and ii) it probabilistically infers entity relationships based on uncertain links using constraint-satisfaction mechanisms. We outline the interface between our scheme and the current data Web, and show how higher-layer applications can take advantage of our approach to enhance search and update of information relating to online entities. We describe a decentralized infrastructure supporting efficient and scalable entity disambiguation and demonstrate the practicability of our approach in a deployment over several hundreds of machines.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; D.2.12.a [**Software Engineering**]: Interoperability—*Data Mapping*

General Terms

Algorithms, Design

Keywords

Entity Disambiguation, Linked Data, Emergent Semantics, Peer Data Management

1. INTRODUCTION

Until recently, the World Wide Web was a hierarchically organized space separating authoritative content providers from relatively passive information consumers. Today, the organization of the World Wide Web has flattened, empowering end-users with new roles. Publishing data on the Web is easier than ever with the advent of new declarative formats like XML, RDF, or Microformats allowing user-defined information to be encoded in machine-processable ways.

With an increasing amount of entities getting created online comes the pressing need to relate and integrate similar entities published by different end-users. Several initiatives, such as the Linked Data movement[1] recently suggested the use of various declarative links to connect semantically related online entities. As a result, a dynamic and decentralized Web of interlinked data is currently emerging on the Internet. We argue in the following that in this new context, entity disambiguation on the Web is drifting from local, pairwise data integration to large-scale, distributed and uncertain social data management.

Let us consider personal identity management as an example. Several formats encoding personal profiles in (semi) structured ways are today getting widely popular. FOAF [6] (an acronym of Friend of a Friend), as an example, is an RDF vocabulary specification describing persons, their activities, and their relations to other people and objects. XML vCard[2] and hCard [2] are two other examples of standards used to encode personal information in semi-structured ways. Structured profiles encoded in proprietary formats can also be found on an increasing list of Web portals or social communities such as DBLP, Wikipedia, LinkedIn, or Facebook. To add to the confusion, an ever increasing number of Web sites create new structured profiles by automatically combining or reformatting some of the aforementioned sources. DBpedia[3] and Spock[4] are two recent examples of this trend.

The result is a flurry of online, disparate, and machine-readable profiles. Relating these different profiles in a meaningful way would open the door to distributed, large-scale and automated personal information management. This remains however infeasible in practice, as these profiles often refer to different identifiers for the same identity, erroneously use a single identifier to refer to several different identities, or present fake identities altogether. As an example, an online survey aiming at retrieving online profiles related to Sir Tim Berners-Lee – the famous computer scientist – revealed the following in mid-2008: we found 109 different structured profiles related to Tim Berners-Lee (see Table 1). Three of them seemed to be created by Tim Berners-Lee himself. 53 profiles were managed by third-parties and another 53 profiles were generated automatically by combining several sources. Some contained legitimate and up-to-date information, while others were outdated or even fake. Information contained in these profiles vary from contact details to bibliographic records or project-related data.

A few of the profiles referred to his FOAF identity (*http://www.w3.org/People/Berners-Lee/card#i*), some to his

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

[1] *http://www.w3.org/DesignIssues/LinkedData.html*
[2] *http://www.xmpp.org/extensions/xep-0054.html*
[3] *http://dbpedia.org/*
[4] *http://www.spock.com/*

Table 1: A sample of structured profiles found online when searching for Sir Tim Berners-Lee

	Outdated	UpToDate	Fake
Managed by himself	0	3	0
Managed by 3rd-party	1	12	40
Automatically Generated	16	4	33

FOAF card (*http://www.w3.org/People/Berners-Lee/card*), his websites or Wikipedia profiles, while most crafted their own, new identifier without relating it to any other identifier. Automatically matching these different profiles is a very difficult task, as entity matching and spurious data detection cannot give ideal results in general (see below for a discussion on this point). This situation is generalizable to other individuals, and more broadly speaking to many other forms of online identities.

In the following, we tackle the problem of meaningfully organizing large graphs of digital entities. We propose a radically new departure for online entity disambiguation based on a probabilistic graph analysis of declarative links relating pairs of entities. We tackle two specific problems: entity disambiguation and temporal discrimination. Entity disambiguation is used to relate all online entities alluding to the same referent. It could for example be used in the above example to retrieve all legitimate profiles related to Tim Berners-Lee. Temporal discrimination handles entities at a finer granularity by distinguishing entities pertaining to the same referent but taken at different points in time. It could for instance be used above to retrieve the most up-to-date profile – or a set of recent profiles – related to Tim Berners-Lee.

1.1 Contributions and Outline

The contributions of this paper include:

1. The identification of two key classes of queries related to semi-structured online entity management: entity disambiguation and temporal discrimination.
2. The description of a taxonomy of declarative constructs used to specify the relationship between entities in a decentralized fashion.
3. The formalization of two probabilistic inference problems based on transitive properties of links used to relate entities in the presence of uncertain information.
4. The description of a fully decentralized, shared-nothing infrastructure used to parallelize the entity discrimination process in large scale settings.
5. The performance evaluation of our approach using large data sets distributed over hundreds of machines.

Our scheme takes advantage of both recent advances in automatic matching techniques and people-powered search [24] and has the following highly desirable properties:

User-driven: Disambiguating entities is known to be a very difficult problem. For instance, differentiating an entity referring to *Jie Wu* (the computer scientist) from an entity referring to *Jie Wu* (the *other* computer scientist), or disambiguating *Jie Wu* (the mathematician) from *Jie Wu* (the *other* mathematician) with limited information – e.g., based on their name and profession only – is virtually impossible. In many cases, human attention is required to disambiguate syntactically equivalent information. The same holds when integrating syntactically different but semantically similar entities, for instance when gathering information for newly married individuals whose last names have just changed.

Best-effort: Information quality is one of the crucial issues of the World Wide Web. Inconsistent, conflicting or simply spurious data can easily shroud more accurate information provided by individual sources. Even if cryptographic authentication mechanisms are today proposed to solve all provenance problems on the Web, their adoption is slow and does not anyway resolve the more general trust issues. Realistically, a very large portion of the information found on the web will still come from questionable or unknown sources in the foreseeable future, and that sea of information cannot be neglected. In this context, we believe that probabilistic mechanisms are an absolute necessity today in order to be able to process heterogeneous data in the large. In what follows, we take advantage of all available information relating the entities, but use both constraint satisfaction and trust-aware mechanisms in order to filter out uncertain or erroneous information automatically. This results in a best-effort, probabilistic and self-organizing network of entities where spurious information gets discarded as more trustworthy information is fed into the system.

Decentralized: Decentralization is a fundamental architectural concept of the Internet, both at the logical (end-user space) and at the physical (i.e., infrastructural) layer. Logical decentralization is necessary to enforce open spaces where anyone is able to express himself, offer services or information. Physical decentralization ensures fault-tolerance: the Internet will continue to function even if one of its major geographical components – say the US, Europe or Asia – goes offline. It also promotes scalability by taking advantage of all independent resources offered to the system. Our approach ensures logical decentralization by taking advantage of any possible source, trusted or not, well-known or not, available on the Internet. Furthermore, we provide a highly-efficient and decentralized physical architecture supporting our scheme, and show how to deploy it on large sets of machines running concurrently.

The rest of this paper is structured as follows: we start by giving a review of related work in the following section. Section 3 formally defines the problem we want to tackle, i.e., relating online entities in the large given uncertain information. Section 4 describes a series of simple constructs we propose to declaratively relate various online entities. We introduce probabilistic models for making sense out of large graphs of related entities in Section 5. Specifically, we introduce a constraint-satisfaction framework to detect erroneous links in Section 5.3.1 and a reputation-based trust mechanism to handle spurious sources in Section 5.3.2. A decentralized architecture implementing our scheme is described in Section 6. We give a performance evaluation of our approach in a real system deployment running on several hundreds of machines in Section 7, before concluding.

2. RELATED WORK

Data Linkage: As the Web is increasingly understood as consisting not only of documents but also of structured data, several initiatives have recently been launched to interlink Web data. XFN [7] is an XHTML profile that is used to consolidate entities by relating URIs through *rel:me* links. *OWL:SameAs* is another popular construct used in a similar manner. Other declarative constructs, such as *Foaf:openid* [6] have been proposed to indirectly link various data by relating them to a common identifier. Okkam [5] is an ongoing large-scale project[5] developing an infrastructure that maintains globally unique entity identifiers (*OKKAM Ids*). They intend to develop a complex entity lineage management together with entity disambiguation methods. At this point, none of these approaches take into account the fact that some (most?) online data might be conflicting, spurious, or erroneous. Our approach supersedes the various declarative

[5] *http://www.okkam.org*

constructs listed above with a taxonomy of constructs defined formally. Furthermore, it encompasses a probabilistic framework to infer entity relationships from a graph of such constructs and describes an infrastructure to support our probabilistic framework in large-scale settings.

Entity Resolution: Our problem is directly related to a broad problem known as entity matching or entity resolution (ER). Entity resolution has been studied extensively in the past few years (see [17] or [18] for recent tutorials). In most solutions, a metric is first proposed to capture the similarity between pairs of entities. Machine learning or lexicographic algorithms are then used to determine whether a pair of entities should be matched or not according to the metric.

Several recent pieces of work follow this classical approach in the context of Web data. Raimond *et. al* [23], for instance, investigate methods towards automatically interlinking music-related data sets on the Web. Their method takes into account both the similarities of the web resources using literal matching as well as the similarity of neighboring resources. Ioannou *et. al.* [15] suggest the use of Bayesian networks to disambiguate entities based on related metadata. Jaffri *et al.* [16] recently investigated entity disambiguation in two popular portals (DBLP and DBpedia) and found that a significant percentage of entities were either conflated or incorrectly linked. Dong *et. al* [12] extend reference reconciliation algorithms in the context of complex information spaces such as personal information management applications. The authors take advantage of contextual information (such as contact lists or email addresses) to reconcile entities and propagate the reconciliation to other related entities. Other recent approaches (e.g., Shen *et. al* [26] or Hogan *et al.* [14]) exploit semantic constraints – as opposed to more common syntactic similarities – to consolidate Web data.

All the aforementioned approaches focus on (iteratively) matching pairs of entities while we focus on *graphs* of entities. Our approach concentrates on link-analysis rather than entity-analysis, and as such cannot be used to match pairs of entities. However, it can take advantage of (uncertain) links created by the aforementioned matchers and exploit the properties of the resulting graph to determine whether or not the entities should really be considered as related. Only a few other link-based entity resolution approaches have been proposed in the past, in contexts different from the one we are presently interested in. We discuss these approaches in more detail in Section 7.3.

3. PROBLEM DEFINITION

The problem we want to solve can be formally introduced as follows: a set of sources $s \in \mathcal{S}$ create and share entities $e \in \mathcal{E}$. Each entity is represented by a globally unique identifier such as a URI, IRI (Internationalized Resource Identifier), or XRI (eXtensible Resource Identifier). Additionally, arbitrary information can be attached to the entities like predicates, values etc.

Each entity models a single referent r (e.g., *Tim Berners-Lee – the computer scientist, Tim Berners-Lee – the amateur photographer, the planet Mars*) taken from a set of distinct referents $\mathcal{R}$: $\forall e \in \mathcal{E}\ \exists r \in \mathcal{R} \mid e \models r$. For example, one could write:

http://w3.org/Berners-Lee/card#i $\models$ *Tim Berners-Lee (CS)*

to express the fact that the entity corresponding to the above URL models Tim Berners-Lee – the computer scientist. Referents can evolve over time. We write r^{t1} to denote referent r at time $t1$.

We say that a pair of entities e_1 and e_2 are equivalent, and write $e_1 \equiv e_2$, when the entities they stand for model the same referent: $e_1 \equiv e_2$ iff $\exists r \in \mathcal{R} \mid e_1 \models r \wedge e_2 \models r$. Pairs of entities which do not satisfy this condition are non-equivalent. We say that an entity e_2 postdates another entity e_1 and write $e_2 \succ e_1$ when both model the same referent, but taken at different times: $e_2 \succ e_1$ iff $\exists r \mid e_1 \models r^{t1} \wedge e_2 \models r^{t2} \wedge t2 > t1$.

Using this framework, we want to answer the following two classes of queries, without having access to any information concerning the referents:

q_1) **[*Entity Disambiguation*]** Which are the entities equivalent to a given entity e?

q_2) **[*Temporal Discrimination*]** Which are the entities postdating a given entity e?

Those queries cover several recurring problems related to online entity management. Query q_1 would certainly be the most frequent query posed in practice, for example to retrieve or integrate all data corresponding to a given online entity. Query q_2 is useful to retrieve the most recent data, to automatically update data, or to keep track of changes related to online entities.

4. IDMESH CONSTRUCTS

To answer the two aforementioned queries, we introduce a taxonomy of simple, declarative constructs that agents can use in order to relate various entities. We introduce the equivalence relation $e_1 \equiv e_2$ and its counterpart, the non-equivalence relation $e_1 \not\equiv e_2$ to allow the sources to express the fact that two entities are equivalent or non-equivalent. Similarly, we introduce the predates ($\prec$), postdates ($\succ$), and equidates ($\|$) constructs to express a relation between a pair of equivalent entities, but modeling their referent at various points of time ($t1 < t2$, $t1 > t2$ and $t1 = t2$ respectively). As they relate entities corresponding to the same referent, those three constructs all imply equivalence relations, i.e., $e_1\{\prec, \succ, \|\}e_2 \Rightarrow e_1 \equiv e_2$. Hence, we actually have to consider a taxonomy of constructs relating entities at two levels of granularity.

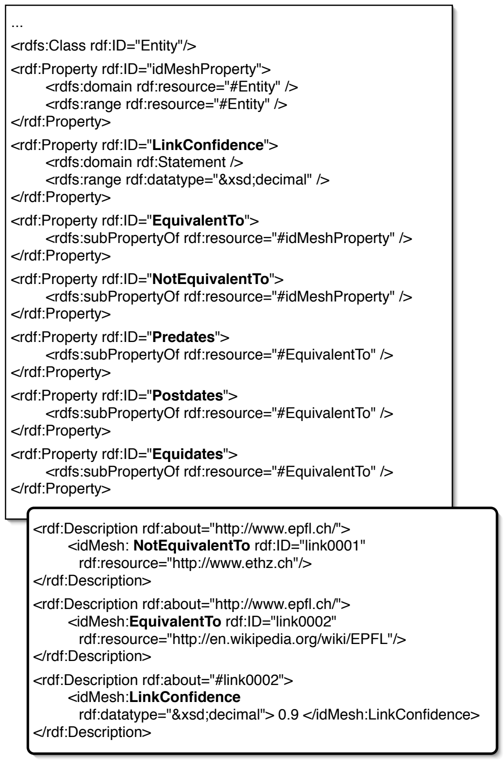
In a perfect world, agents managing some entities would relate the identifiers they use to represent their entities to related identifiers using the above constructs and thus explicitly answer the queries posed above. Realistically on the Internet, however, a potentially large fraction of relations will be missing, uncertain or even erroneous: with the vibrant activities relating to mash-ups and automatic entity matching, we can expect the majority of relations to be created by software agents or programs, usually with a certain confidence value. Furthermore, some human or software agents might decide to lure the system for their own benefits (phishing, spamming etc.) by voluntarily entering erroneous information. Finally, as individuals do not always agree, have limited knowledge, and sometimes make mistakes, incorrect or contradicting relations might be entered by legitimate agents.

We capture the uncertainty related to the relation between two entities with a confidence value c. Writing $e_1 \equiv_c e_2$ expresses a probabilistic equivalence between e_1 and e_2: $P(e_1 \equiv e_2) = c$. In the following, we suppose that certain relations are implied when confidence values are omitted, e.g, that $e_1 \equiv e_2$ and $e_1 \equiv_{1.0} e_2$ encode the same information.

Figure 1 shows how to express the different constructs introduced above using an XML serialization of RDF. Note that in a large-scale deployment, different formats and already existing constructs (such as XFN *rel:me* links or *OWL:sameas* constructs) could easily be integrated into this picture by taking advantage of syntactic wrappers and decentralized integration techniques [11].

5. MAKING SENSE OUT OF IT

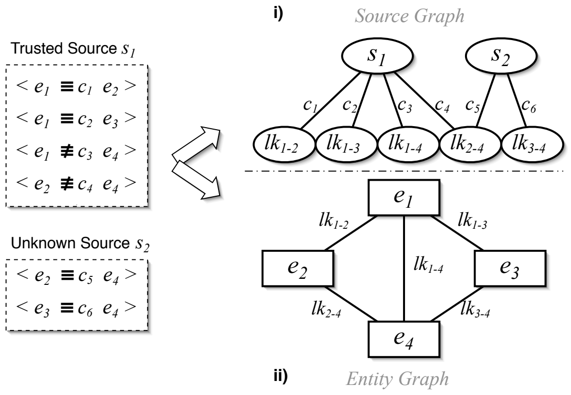
In the following, we study how to make sense out of the various probabilistic links we have just described. We introduce a probabilistic inference problem based on graphs of related entities. We start by a brief example giving an intuitive idea of our method and an overview of factor-graphs before delving into the core of our approach.



```
...
<rdfs:Class rdf:ID="Entity"/>
<rdf:Property rdf:ID="idMeshProperty">
        <rdfs:domain rdf:resource="#Entity" />
        <rdfs:range rdf:resource="#Entity" />
</rdf:Property>
<rdf:Property rdf:ID="LinkConfidence">
        <rdfs:domain rdf:Statement />
        <rdfs:range rdf:datatype="&xsd;decimal" />
</rdf:Property>
<rdf:Property rdf:ID="EquivalentTo">
        <rdfs:subPropertyOf rdf:resource="#idMeshProperty" />
</rdf:Property>
<rdf:Property rdf:ID="NotEquivalentTo">
        <rdfs:subPropertyOf rdf:resource="#idMeshProperty" />
</rdf:Property>
<rdf:Property rdf:ID="Predates">
        <rdfs:subPropertyOf rdf:resource="#EquivalentTo" />
</rdf:Property>
<rdf:Property rdf:ID="Postdates">
        <rdfs:subPropertyOf rdf:resource="#EquivalentTo" />
</rdf:Property>
<rdf:Property rdf:ID="Equidates">
        <rdfs:subPropertyOf rdf:resource="#EquivalentTo" />
</rdf:Property>
```

```
<rdf:Description rdf:about="http://www.epfl.ch/">
        <idMesh: NotEquivalentTo rdf:ID="link0001"
          rdf:resource="http://www.ethz.ch"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.epfl.ch/">
        <idMesh:EquivalentTo rdf:ID="link0002"
          rdf:resource="http://en.wikipedia.org/wiki/EPFL"/>
</rdf:Description>
<rdf:Description rdf:about="#link0002">
        <idMesh:LinkConfidence
          rdf:datatype="&xsd;decimal"> 0.9 </idMesh:LinkConfidence>
</rdf:Description>
```

Figure 1: The *id*Mesh constructs expressed in an XML serialization of RDF, along with an example expressing the relations between three entities corresponding to two Swiss institutes of technologies.



i)
Source Graph
Trusted Source s_1
$< e_1 \equiv c_1 \; e_2 >$
$< e_1 \equiv c_2 \; e_3 >$
$< e_1 \not\equiv c_3 \; e_4 >$
$< e_2 \not\equiv c_4 \; e_4 >$
Unknown Source s_2
$< e_2 \equiv c_5 \; e_4 >$
$< e_3 \equiv c_6 \; e_4 >$
s_1
s_2
c_1 c_2 c_3 c_4 c_5 c_6
lk_{1-2} lk_{1-3} lk_{1-4} lk_{2-4} lk_{3-4}
e_1 e_2 e_3 e_4
ii)
Entity Graph

Figure 2: An introductory example where two sources relate four entities; two graphs can be derived from the statements on the left: a source graph relating entity links to their sources (i) and an entity graph capturing the relations between entities (ii).

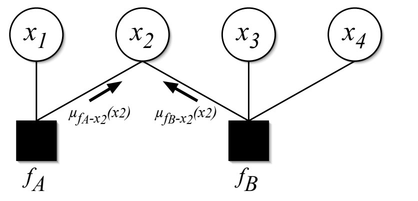


Figure 3: A simple factor-graph of four variables and two factors

5.1 An Introductory Example

Figure 2 depicts a simple example where two sources, s_1 and s_2, relate four entities e_1 to e_4 by defining equivalence links lk between the entities with various confidence values c. We can create two graphs from the links defined by the sources: a bipartite *source graph* relating sources to the links they define with their corresponding confidence values (Figure 2 i) and an *entity graph* depicting how entities are related through the links (Figure 2 ii). Let us suppose that source s_1 is a trusted source (e.g., a well-known source, which published his links on an openID-enabled website), while s_2 is an unknown malicious source. We make two observations by analyzing the graphs we have just introduced. First, we can process the links differently depending on their sources. For instance, the two links defined by s_2 should be considered with some care as they are defined by an unknown source. Second, we can infer additional links by taking advantage of the fact that the equivalence relation is symmetric and transitive. For instance, $(e_1 \equiv e_2) \wedge (e_2 \equiv e_4) \Rightarrow (e_1 \equiv e_4)$.

Our probabilistic algorithm systematizes these two observations to determine the most probable relation between pairs of entities. For the network of Figure 2 for example, it first detects a conflict for lk_{2-4}, which is defined as non-equivalent by s_1 and equivalent by s_2. Other conflicts are automatically detected, such as for the links between e_1, e_3, and e_4: s_1 defines e_1 and e_3 as equivalent and e_1 and e_4 as non-equivalent, while s_2 declares e_3 and e_4 as equivalent, which is impossible ($(e_1 \equiv e_3) \wedge (e_3 \equiv e_4)$ should imply $(e_1 \equiv e_4)$, which is not the case as s_1 defines e_1 and e_4 as non-equivalent). Taking into account all possible observations, our algorithm lowers the confidence of the links declared by s_2 and concludes that e_1, e_2, and e_3 are equivalent and different from e_4. We explain the details of our algorithm in the following.

5.2 A Quick Reminder on Factor-Graphs and Message Passing Schemes

We use factor-graphs to graphically represent probabilistic variables and distributions in the following. Note that our approach is not bound to this representation – we could use series of conditional probabilities only or any other probabilistic graphical model – but we decided to use factor-graph for their illustrative merits.

We give below a brief introduction to factor-graphs and 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 [19]. 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.,

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

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 simple *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. 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 [21]. 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 modeling entity graphs.

5.3 Deriving a Factor-Graph to Retrieve Equivalent Entities

We start by answering the first query on entity equivalence by taking advantage of equivalent and non-equivalent relations defined by the sources. We also indirectly take advantage of the predates, postdates and equidates constructs which are defined as specialized cases of equality constructs (see Sections 3 and 4). To answer this query, we derive a probabilistic network from the entity and the source graphs (see Figure 4). We describe this probabilistic network in two steps: first its lower part, related to constraint satisfaction, and then its upper part, which handles reputation-based trust management.

5.3.1 Graph-Based Constraint Satisfaction

We start by defining a graph-based constraint satisfaction problem from the entity graph. We define constraints by taking advantage of the symmetry and the transitivity of the equivalence relations. We introduce binary variables lk representing the equivalent and non-equivalent links relating the entities. Link variables lk can take values eq or $noneq$ depending on whether they represent equivalence or non-equivalence relations. Obviously, $P(lk = eq) + P(lk = noneq) = 1$ for our link variables. The initial values of these variables can be defined by prior density functions.

Examining the graph of related entities, we observe that series of equivalent and non-equivalent links form cycles, for example by going from e_1 to e_2, e_4, and back to e_1 by following lk_{12}, lk_{24}, and lk_{41} in Figure 4 ii. The equivalence relation we have defined in Section 3 is symmetric: $\forall e_1, e_2 \in \mathcal{E},\ e_1 \equiv e_2 \Rightarrow e_2 \equiv e_1$. It is also transitive: $\forall e_1, e_2, e_3 \in \mathcal{E},\ e_1 \equiv e_2 \wedge e_2 \equiv e_3 \Rightarrow e_1 \equiv e_3$. As two entities cannot be at the same time equivalent and non-equivalent, we observe that no cycle can contain exactly one non-equivalent link: $n-1$ equivalent links in a cycle of n links $lk_1, \ldots, lk_n$ obligatory imply – by symmetry and transitivity – that the last link is equivalent as well. Thus, we pose a graph constraint $gc()$ for each cycle discovered in the graph to forbid variable assignments where exactly one link is non-equivalent in the cycle. The graph constraint function $gc()$ relating links $lk_{12}, \ldots, lk_n$ can be defined in a compact form by the the following conditional probability function:

$$P(gc = 1 | lk_1, \ldots, lk_n) = \begin{cases} 0 & \text{if exactly one noneq} \\ 1 & \text{otherwise} \end{cases}$$

and by fixing the constraint variable gc to 1 to rule out impossible assignments of $lk_1, \ldots, lk_n$ variables.

This conditional probability function allows us to define a global constraint satisfaction factor-graph from a network of interconnected entities. We create the factor-graph by linking variables representing the equivalence links to constraint functions whenever a link is part of a cycle in the graph. The algorithm to derive the constraint satisfaction factor-graph is given in Algorithm 1. The lower part of Figure 4 iii) – consisting of link variables lk together with their prior distributions and their graph constraint functions $gc()$ – gives an example of such a factor-graph. There are three constraint functions in this graph, as three cycles can be identified from the entity graph.

Algorithm 1 Deriving a constraint satisfaction factor-graph

```
/*create variable and prior factor for each lk link*/
for all link lk in entity graph do
    add lk.factor to constraint-factor-graph;
    add lk.variable to constraint-factor-graph;
    connect lk.factor to lk.variable;
end for
for all cycles c in entity graph do
    /*create factor representing constraint for each cycle*/
    add gc.factor to constraint-factor-graph;
    for all link lk in cycle c do
        /*connect link variables to cycle constraints*/
        connect gc.factor to lk.variable;
    end for
end for
```

The constraint factor-graphs allow us to detect inconsistencies in the network of entities by inferring posterior probabilities for the lk variables through iterative sum-product operations (see Section 5.2 above, and the performance evaluation in Section 7). Note that the cycles are typically *not* independent of each other in this setting: two cycles are correlated as soon as they share one equivalence link. Thus, local updates on the entity graph (e.g., new equivalence link) can have repercussions on distant variables in the factor-graph.

5.3.2 Reputation-Based Trust Management

After having defined a constraint satisfaction factor-graph, we define a reputation-based trust management factor-graph for the sources $s \in \mathcal{S}$ providing equivalence/non-equivalence links based on the source graph.

Our goal is this time to maintain probabilistic trust variables t attached to the different sources s. We define trust variables as taking value $trusted$ if the corresponding source is trustworthy (i.e., provides correct relations between the entities) and $untrusted$ otherwise. Each trust variable has a prior distribution $t()$ capturing the initial degree of trust for this source. For example, trust variables can be initialized to high $trusted$ values when agents operate within a closed domain (e.g., in the *http://www.mit.edu* domain) or when they can be authenticated and are well-known. Other mechanisms, such as reputation-based trust mechanisms from P2P networks [1], or trust metrics for online communities [8] can also be used to initialize the trust variables. In case no information is available for some source, prior distributions are initialized to $\{0.5, 0.5\}$ by default (maximum entropy principle).

The trust management part of the factor-graph serves three purposes: it takes into account external trust values as described above, it updates the trust variables whenever conflicts are detected by the constraint satisfaction graph, and ponders the confidence values attached to the links by taking into account the trust variables of the

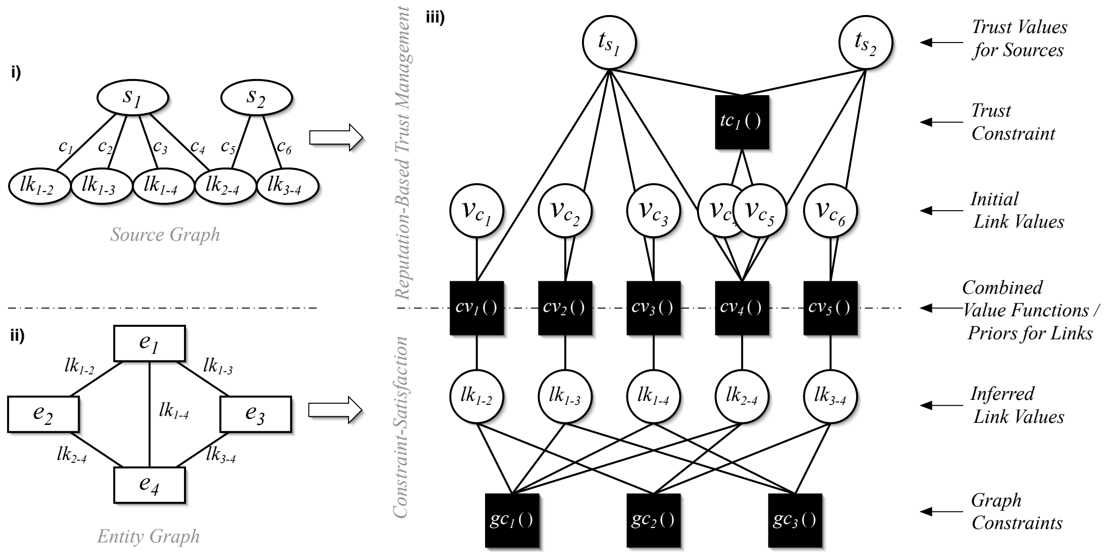


Figure 4: Two sources declaring several links (i) to relate four entities (ii), resulting in a probabilistic network (iii), handling both graph-based constraint satisfaction (lower part of iii) and reputation-based trust management (upper part of iii).

sources. We introduce initial value variables v taking values eq when a link is considered as being equivalent and $noneq$ otherwise. The prior distribution of these variables is initialized with the confidence values c specified by the sources: $\{c, 1-c\}$ for equivalence links and $\{1-c, c\}$ for non-equivalence links. For a list of sources $s_1, \ldots, s_n$ with trust variables $t_1, \ldots, t_n$ proposing initial values $v_1, \ldots, v_n$ for a given link, we compute a combined value for the link using the following conditional function $cv()$:

$$P(cv = eq|\{(t_1, \ldots, t_n), (v_1, \ldots, v_n)\})$$
$$= \begin{cases} \frac{\sum_i \mathbb{1}_{(v_i = eq \ \wedge \ t_i = trusted)}}{\sum_i \mathbb{1}_{(t_i = trusted)}} & \text{if } \exists t_i | t_i = trusted \\ 1/2 & \text{otherwise} \end{cases}$$

where $\mathbb{1}_{(cond)}$ is an indicator function equal to 1 when $cond$ is true and 0 otherwise, and where cv is a binary variable taking values eq or $noneq$.

The combined value function $cv()$ combines the different initial values by systematically considering the values proposed by untrustworthy sources as being less important than values proposed by trustworthy sources. For example, for only one source s with a probability of being trustworthy p_t suggesting an initial value for a link with a probability c of being equivalent, the trust function gives as output $P(cv = eq) = p_t * c + 1/2(1 - p_t)$. That is, it reduces the values proposed by untrustworthy sources to utterly uncertain values ($P(cv = eq) = 1/2$ if $p_t = 0$), while keeping the values of trustworthy sources intact ($P(cv = eq) = c$ if $p_t = 1$).

In addition, we define constraints to forbid impossible value assignments for $t_1, \ldots, t_n$ and $v_1, \ldots, v_n$. As two entities cannot be at the same time equivalent and non-equivalent, we rule out impossible assignments when conflicting values (i.e., both equivalent and non-equivalent values) are simultaneously considered by trusted sources using the following conditional probability function $tc()$:

$$P(tc = 1|\{(t_1, \ldots, t_n), (v_1, \ldots, v_n)\})$$
$$= \begin{cases} 1 & \text{if } v_i = v_j \ \forall (v_i, v_j) \mid t_i = t_j = trusted \\ 0 & \text{otherwise} \end{cases}$$

and by fixing the constraint variable tc to 1. This constraint would for example rule out cases where two trustworthy sources ($t_1 = t_2 = trusted$) simultaneously consider a link as being respectively equivalent ($v_1 = eq$) and non-equivalent ($v_2 = noneq$). Note that a similar constraint could be defined at the entity graph level (see preceding section) by considering a multigraph model for the entity graph and analyzing cycles of length two, at the expense of producing much bigger probabilistic models.

We construct the factor-graph corresponding to the trust part by producing one combined value factor $cv()$ for each link in the source graph, by connecting trust variables t and initial value variables v to those factors for each source declaring a relation for the link, and by adding trust constraints factors $tc()$ whenever appropriate (see Algorithm 2).

Algorithm 2 Deriving a trust factor-graph from a source graph

```
/*add combined value factor cv() for each link lk*/
for all link lk in source graph do
    add cv.factor to trust-factor-graph;
    /*add trust variable and initial value variables*/
    for all source s with trust variable t connected to lk in source
    graph with initial value v do
        add t.variable to trust-factor-graph;
        connect cv.factor to t.variable;
        add v.variable to trust-factor-graph;
        connect cv.factor to v.variable;
    end for
    /*add trust constraints */
    if cv.factor.neighbors() $> 2$ then
        add tc.factor to trust-factor-graph;
        for all neighbor in cv.factor.neighbors() do
            connect tc.factor to neighbor;
        end for
    end if
end for
```

5.3.3 Putting It Altogether

Finally, we connect the trust factor-graphs to the constraint satisfaction factor-graph by replacing the prior values of the link variables lk by the values cv given as output by the combined value functions $cv()$ (see Figure 4). This connection generates more accurate prior values for the link variables lk in the constraint satisfaction factor-graph. These values are used in the constraint satisfaction problem to infer more plausible values for the link variables. On the trust factor-graph side, lk variables are now fed to combined value functions $cv()$ as cv values. These values, derived from the constraint-satisfaction part, influence the trust values t of the agents by lowering the trustworthiness of the agents proposing very improbable initial values v from a constraint-satisfaction perspective. Thus, we create an autocatalytic, reinforcement process where constraint-satisfaction helps discovering untrustworthy sources and where trust management delivers in return more reasonable prior values for the link variables.

5.4 Deriving a Factor-Graph to Retrieve Up-to-date Entities

To answer the second query we analyze the predate, postdate, and equidate relations defined by the sources. This case can be seen as a generalization of the equivalence case described above. Links are this time defined as ternary variable taking value pre, $post$, or $equi$. The initial values are also mapped onto ternary variables. For example, if a link is described as $lk = pre$ with confidence c, the prior distribution of initial value v would be $(c, \frac{1-c}{2}, \frac{1-c}{2})$, where the first element corresponds to the probability of the initial value being pre, the second to being $post$ and the third to being $equi$. We have chosen this model for simplicity but other models such as Dempster-Shafer belief functions [25] could be used as well.

It should be noted that the links corresponding to pre and $post$ are directed. As for the equivalence case, constraints arise because of the transitivity properties of the three relations. As an example, if $e_1 \prec e_2$ and $e_2 \prec e_3$ then $e_1 \prec e_3$. As a result, no directed cycle can have a single $equi$ link while all the other links are either pre or $post$. Other constraints can be similarly inferred. We pose the directed cycle constraint function $gc()$ relating links $lk_1, \ldots, lk_n$ which form a cycle as:

$$P(gc = 1 | lk_1, \ldots, lk_n) = \begin{cases} 0 & \text{if exactly one equi, others either pre or post} \\ 0 & \text{if exactly one pre or post, others equi} \\ 0 & \text{if all pre or all post} \\ 1 & \text{otherwise.} \end{cases}$$

By fixing the constraint variable gc to 1 impossible assignments of the $lk_1, \ldots, lk_n$ variables are ruled out. The trust-related derivations are similar to the equivalence case, with ternary variables for the initial values v.

5.5 Query Answering

Once the factor-graphs defined above have been built, answering the two queries from Section 3 is straightforward. Running an inference algorithm on the factor-graphs creates posterior values for the link variables lk. Query q_1 can for example be answered by starting to crawl the entity graph at entity e and returning the set of entities e' encountered when following all links lk such that $P(lk = eq) > P(lk = noneq)$. Query q_2 can be answered in a similar manner, by following all directed links such that $P(lk = pre) > P(lk = equi) \wedge P(lk = pre) > P(lk = post)$ and by backtracking on all links such that $P(lk = post) > P(lk = equi) \wedge P(lk = post) > P(lk = pre)$.

6. SYSTEM PERSPECTIVE

The scheme proposed above defines a probabilistic model with a potentially very large number of entities originating from a flurry of online sources. Storing entities and answering queries on a single machine would limit the size of the entity graphs that can be handled. In the following, we give a succinct description of *id*Mesh, a highly parallel, scalable, asynchronous, and shared-nothing peer data management infrastructure tackling the issue of distributing our problem over large sets of machines simultaneously.

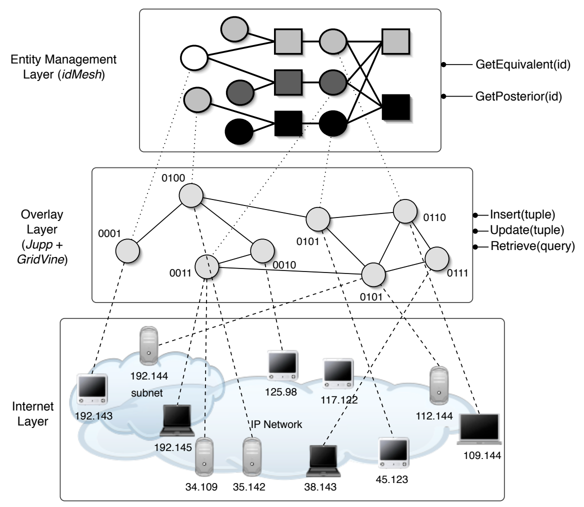


Figure 5: The *id*Mesh system takes advantage of an overlay layer (middle layer) to manage a decentralized probabilistic entity graph (top layer)

6.1 Architectural Overview

The design of our system builds on some of our previous work on peer data management infrastructures [11] and decentralized probabilistic networks for schema mappings [10]. The general architecture of our system is given in Figure 5. *id*Mesh is based on a structured overlay built on top of the Internet. The overlay layer (middle part of Figure 5) creates and maintains a load-balanced identifier space partitioned over the peers. We take advantage of this identifier space to share identifiers and semi-structured data relating to entities using the GridVine [11] peer data management system. All data is kept in local databases at the peers, but indexed globally using the identifier space of the overlay network. Data can be fed into the system by Web crawlers or created by the participating peers directly depending on the application.

6.2 Distributed Probabilistic Inference

In addition to inserting and correctly indexing tuples pertaining to the entities and the entity links, the peers collaboratively create a factor-graph to infer equivalence and postdate relations between entities. The peer responsible for inserting/updating a link is also responsible for creating probabilistic nodes (variables, factors) and edges related to this link. It relies on the global index whenever necessary, for example when searching for link cycles to create the constraint part of the factor-graph. The roles of the probabilistic nodes are to receive messages, to compute values using functions, and to send updated messages to other nodes as described in the preceding section in order to infer values for the entity links.

We use the identifier space maintained by the overlay layer to distribute the probabilistic graph. The simplest solution to map the probabilistic graph to the identifier space is to handle all node separately by creating an identifier for each node and placing it at the corresponding peer(s) in the identifier space. This however creates significant network traffic between the peers: as explained in Section 5.2, each round of the inference process requires two messages per edge in the factor-graph. For instance, our example factor-graph (Figure 4) would then generate 62 messages per inference round. We have thus to find the right tradeoff between distribution and centralization of the probabilistic nodes over the identifier space. Distribution is desirable to parallelize the workload using as many peers as possible and to provide better fault tolerance. Centralization is compelling for reducing network traffic and minimizing the number of updates sent to the peers when a link is created, updated, or deleted.

Our approach partitions the factor-graph by regrouping logically related nodes to reduce network traffic while keeping the number of updates triggered by link creation/deletion small and ensuring a good degree of distribution on the overlay. We regroup all nodes directly pertaining to a link (i.e., the inferred link value, the value function, and the initial link value) at one point in the overlay. We also regroup the graph constraints and the trust constraints with one of the links / sources they are related to. In this manner, we reduce the number of network messages required for an inference round on our example graph from 62 messages to 26 messages only. An extensive analysis of the partitioning trade-off is outside of the scope of this paper. However, we experimentally show in Section 7 that our partitioning provides a good parallelization of the problem in a P2P deployment.

7. PERFORMANCE EVALUATION

We give in the following a performance evaluation of our approach deployed within the *id*Mesh system. The system has many parameters and dimensions. We focus below on what we believe are the most important results and metrics to capture the general behavior of our system. Specifically, we focus on the accuracy of the inference based on the constraint graph, the trust graph, or both, and on the scalability of our system. Because of space constraints, we focus on entity disambiguation factor-graphs only. Constraint-satisfaction factor-graphs for temporal discrimination are structurally similar and exhibit the same properties.

7.1 Performance of the Inference Network

7.1.1 *Graph-Based Constraint-Satisfaction*

We start by studying the accuracy of the inference based on the constraint-satisfaction part only. For the experiments, we create networks of i entities, split into $i/10$ groups of equivalent entities. We create l equivalence/non-equivalence links by randomly selecting pairs of entities. For this experiment, we set the prior distributions of the links to $(0.9, 0.1)$ for the links relating equivalent entities, and to $(0.1, 0.9)$ for the links relating non-equivalent entities. These values are swapped for erroneous links, which represent a variable fraction of the links. The results presented hereafter are averaged over 20 consecutive runs with confidence intervals at 95%.

One important parameter for the graph-based constraint satisfaction is the density of links in the entity graph. This density has an influence on the number of cycles in the graph, and thus on the number of constraints. Social networks tend to contain 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 [4]); the longer the cycle, however, the less interesting it is from an inference point of view as it relates to a higher number of variables (and hence represents less precise information). Figure 6 shows the effect of considering smaller or bigger loops for networks with various link densities. We show on the graph the

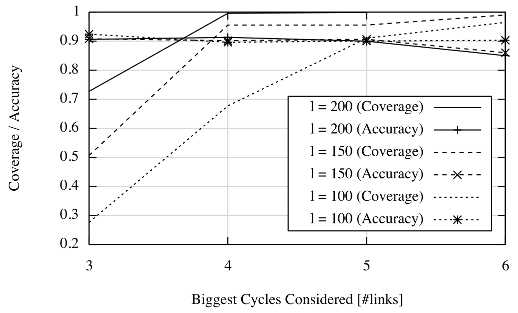


Figure 6: Inference accuracy and coverage for the graph-based constraint-satisfaction, for networks of 50 entities, 100/150/200 links, 10% erroneous links, and varying cycle length max.

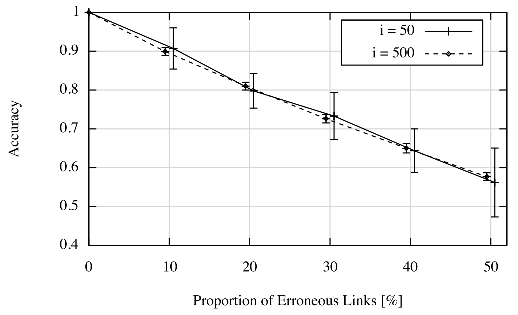


Figure 7: Inference accuracy for the graph-based constraint-satisfaction, for networks of 50 and 500 entities, 150/3000 links, and a varying fraction of erroneous links

coverage of the inference, defined as the fraction of links that are taken into account by the inference (i.e., fraction of links that are in at least on cycle) and the accuracy of the inference, defined as the fraction of links inferred correctly.

The accuracy of the inference basically does not depend on the length of the cycles considered. For dense networks ($l = 150$ or $l = 200$), however, considering longer cycles presents a slight disadvantage because of the reason explained above. Considering longer cycles is however beneficial for the coverage of the inference, especially for sparse networks where few shorter cycles exist. In the following, we consider relatively dense networks and cycles up to size 4.

Figure 7 shows the accuracy of the graph-based constraint-satisfaction inference for a varying fraction of erroneous links from 0 to 50%. Quite naturally, the more erroneous links, the more difficult it is for the constraints to determine which links are equivalent or and which are not. Note that taking different values for the priors (e.g., 0.8 or 0.7) does not change the results significantly. Note also that the size of the graph has no impact on the accuracy of the inference: Figure 7 considers two networks, one with 50 entities and 150 links and a second one with 500 entities and a number of links (3000) chosen to get a density of cycles similar to the first network.

7.1.2 *Trust Management*

We now turn our attention to the trust management part of the graph by discarding all cycles in the entity graph. We consider three simple classes of sources for illustration purposes: *legitimate* sources, which are trusted ($P(t = trusted) = 1$) and define cor-

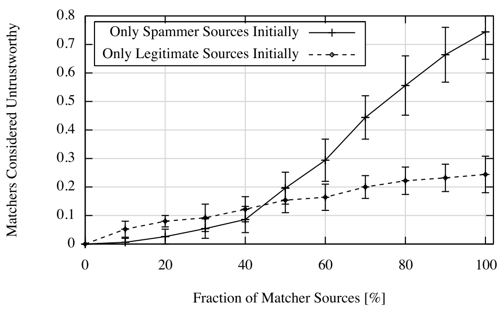


Figure 8: Fraction of matchers inferred as untrustworthy ($P(t = trusted) < 0.25$) by the trust management part in a network of 50 entities, 150 links, 50 sources each declaring values for 1-10 links, and a growing fraction of matchers

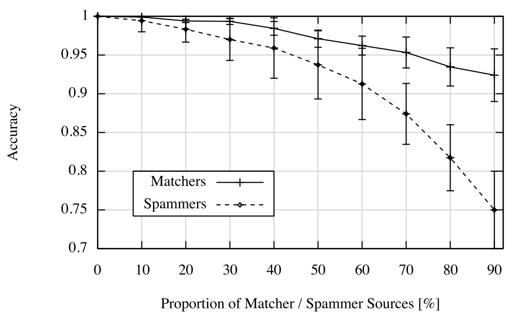


Figure 9: Inference accuracy for networks of 50 entities, 150 links, 50 sources each declaring values for 1-10 links, and an initial population of legitimate sources gradually replaced by matchers or spammers.

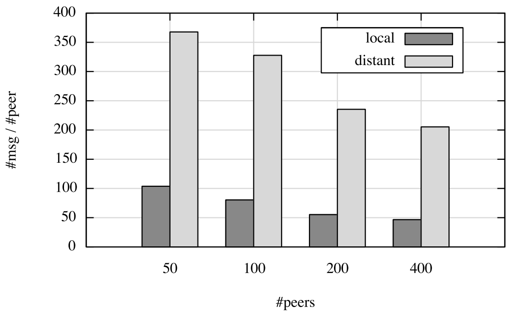


Figure 10: Average number of local and distant messages sent by a peer to solve inference, starting with 50 peers, 1000 entities, 3000 links and 100 sources, and doubling all values at every step.

rect links with confidence $c = 1$. Automatic *matcher* sources, which are initially *unknown* to the system ($P(t = trusted) = 0.5$), define correct links 90% of the time, and give confidence $c = 0.9$ to their links. *Spammer* sources, finally, are unknown as well (trust $P(t = trusted) = 0.5$), define incorrect links, and give confidence $c = 1.0$ to their links. Note that our approach can handle more complex behaviors (e.g., occasional cheaters) in a comparable manner. We create random networks of 50 entities interlinked by 150 links. We then create 50 sources, and let each source define a value for 1 to 10 of the links.

The goal of the trust management part is to determine which sources can be trusted and which cannot. For populations consisting of a mix of legitimate and spammer sources only, the algorithm can always determine which sources can be trusted. Things are more complex when matchers are present, as matchers sometimes propose correct values, and sometimes not. Figure 8 shows the fraction of matchers considered as highly untrustworthy (inferred trust $P(t = trusted) < 0.25$), starting with a population of spammers or a population of legitimate sources initially, and gradually replacing a fraction of the sources by matchers. Matchers add here uncertainty to the trust process. In case no source can be trusted initially (e.g., no legitimate source initially), matchers get increasingly considered as untrustworthy as more and more conflicting evidences emerge.

7.1.3 *Combined Analysis*

Figure 9, finally, gives the accuracy of our approach at inferring equivalence and non-equivalence relations for the links when integrating trust management and constraint satisfaction. The graph considered has 50 entities, 150 links, 50 sources each declaring values for 1 to 10 links. All sources are legitimate initially, and then are gradually replaced by matchers or spammers. We observe that the resulting inference process is very resilient: for example, it successfully discovers the relations between entities with an accuracy of 75% even when 90% of the sources are actually spammers feeding erroneous information into the system.

7.2 System Scale-Up

All operations related to the factor-graph are handled in a decentralized and asynchronous way by the *id*Mesh platform. Inference is handled by sending local messages between the probabilistic nodes belonging to the same partition, and by sending P2P messages between different partitions. A probabilistic node sends an update messages to its neighbors whenever it receives at least half of the update messages it is expecting from the other nodes. It stops sending updates whenever the new value it has to send is within 5% of the two previous values it has already sent. Implicitly then, the inference process stops when all probabilistic nodes decide to stop sending updates.

To measure the scalability of our system, we consider both local messages sent from one probabilistic node to another probabilistic node on the same peer, and distant messages sent to another partition and requiring P2P routing. We start with a network of 50 peers, 1000 entities, 3000 links and 100 sources, and double all values to generate scale-up versions of the graphs for 100, 200, and finally 400 peers. In this setting, each link receives a value from two sources. The tests were conducted on a local area cluster with 50 to 400 machines (we run one peer per machine), and averaged over 5 runs. As can be seen in Figure 10, the number of messages generated scales sub-linearly with the size of our problem, mainly due to the rarefaction of short cycles in the larger networks we consider.

7.3 Comparison With Previous Approaches

As discussed in Section 2, classical entity resolution mechanisms focus on entity-analyses and cannot be used in link-only settings such as those described above. To the best of our knowledge, all related link-based entity resolution approaches were developed in contexts anterior to the linked data emergence and would be ill-suited to our setting as well, since they do not explicitly take into account the uncertainty related to the links. Methods based on geodesic (i.e., path-based) distances in the entity graph such as [13] or [20] cannot discriminate uncertain links based on network dispersed evidences. They would thus either discard uncertain information from the matchers and spammers, or treat uncertain information on a uniform basis, generating poor results in both cases (low precision stemming respectively from low coverage and erroneous confidence values for the links). Graph partitioning and clustering methods such as [3] or [22] would suffer from the same fundamental problems, in the sense that they would not be able to disambiguate the links used to analyze the structure of the graph without the ground-truth related to the trust value of the sources.

In the end, the distinctiveness of our approach lies in the application of two core Emergent Semantics [9] principles that cannot be emulated by previous approaches: the analysis of the transitive closures of the probabilistic links, and the reinforcement of global information through network dispersed local evidences.

8. CONCLUSIONS

As the data Web develops, managing heterogeneous online entities is becoming a key problem impeding online data processing and information reuse. Current approaches mostly focus on matching pairs of entities, either by asking the help of end-users or by creating automatic matchers. We proposed in this paper a different approach, based on an analysis of graphs of interlinked entities. Our method complements previous approaches, and could be used in combination with them (in fact, the OKKAM project is investigating such possibilities). Our approach leverages entity relationships to identify constraints and to resolve conflicts by handling trust metrics attached to the sources declaring the relationships.

The technique we presented can be extended in many ways. One compelling extension would be to generalize the constructs we defined to answer other classes of queries. An interesting example would be the *relatedness* relationship. The semantics of this relationship are not well defined in general, but could be expressed in many specific contexts, for example for several variations of rel tags or FOAF links.

9. ACKNOWLEDGEMENT

The work presented in this paper was partially supported by the European project OKKAM No. 215032.

10. REFERENCES

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *International Conference on Information and Knowledge Management (CIKM)*, 2001.

[2] J. Allsopp. *Microformats: Empowering Your Markup for Web 2.0*. Friends of ED Publisher, 2007.

[3] R. Bekkermand and A. McCallum. Disambiguating Web Appearances of People in a Social Network. In *International World Wide Web Conference (WWW)*, 2005.

[4] G. Bianconi and M. Marsili. Loops of any size and hamilton cycles in random scale-free networks. *Journal of Statistical Mechanics: Theory and Experiments*, P06005, 2005.

[5] P. Bouquet, H. Stoermer, C. Niederee, and A. Mana. Entity Name System: The Backbone of an Open and Scalable Web of Data. In *IEEE International Conference on Semantic Computing (ICSC)*, 2008.

[6] D. Brickley and L. Miller. FOAF Vocabulary Specification 0.91. http://xmlns.com/foaf/spec/.

[7] T. Celic, M. Mullenweg, and E. Meyer. Xhtml Friends Network Relationships Meta Data Profile 1.1. http://gmpg.org/xfn/11.

[8] H. Choi, S. Kruk, S. Grzonkowski, K. Stankiewicz, B. Davis, and J. Breslin. Trust Models for Community-Aware Identity Management. *Identity, Reference, and the Web Workshop at the World Wide Web Conference (WWW)*, 2006.

[9] P. Cudré-Mauroux. *Emergent Semantics*. EPFL & CRC Press, 2008.

[10] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic Message Passing in Peer Data Management Systems. In *International Conference on Data Engineering (ICDE)*, 2006.

[11] P. Cudré-Mauroux, S. Agarwal, and K. Aberer. Gridvine: An infrastructure for peer information management. *IEEE Internet Computing*, 11(5), 2007.

[12] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD International Conference on Management of Data*, 2005.

[13] R. Hoelzer, B. Malin, and L. Sweeney. Email Alias Detection Using Social Network Analysis. In *International Workshop on Link Analysis (LinkKDD)*, 2005.

[14] A. Hogan, A. Harth, and S. Decker. Performing object consolidation on the semantic web data graph. In *I^3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web workshop at the World Wide Web conference (WWW)*, 2007.

[15] E. Ioannou, C. Niederée, and W. Nejdl. Probabilistic Entity Linkage for Heterogeneous Information Sources. In *International Conference on Advanced Information Systems Engineering (CAiSE)*, 2008.

[16] A. Jaffri, H. Glaser, and I. Millard. URI Disambiguation in the Context of Linked Data. In *Workshop on Linked Data on the Web (LDOW)*, 2008.

[17] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *ACM SIGMOD international conference on Management of data*, 2006.

[18] N. Koudas and D. Srivastava. Approximate Joins: Concepts and Techniques. In *International Conference on Very Large Data Bases (VLDB)*, 2005.

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

[20] E. Minkov, W. Cohen, and A. Ng. Contextual search and name disambiguation in email using graphs. In *International Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[21] 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.

[22] B. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving Grouped-Entity Resolution using Quasi-Cliques. In *IEEE International Conference on Data Mining (ICDM)*, 2006.

[23] Y. Raimond, C. Sutton, and M. Sandler. Automatic Interlinking of Music Datasets. In *International Workshop on Linked Data on the Web (LDOW)*, 2008.

[24] W. Roush. People-powered search. *MIT Technology Review*, May-June, 2007. https://www.technologyreview.com/Infotech/18655/.

[25] G. Shafer. *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.

[26] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, 2005.