

Decentralized Semantic Coordination Through Belief Propagation.*

George A. Vouros

Department of Digital Systems, University of Piraeus, Greece

Abstract. This paper proposes a decentralized method for communities of agents to reach semantic agreements on subjective ontology elements' correspondences / mappings, via belief propagation. Agents detect disagreements on mapping decisions via feedback they receive from others, and they revise their decisions on correspondences with respect to their mapping preferences and the semantics of ontological specifications. We address this problem as a distributed extension of the max-plus algorithm via belief propagation. We show the potential of such an approach in a number of networks with varying complexity and for different cases of semantic disagreement between agents.

1 Introduction

Interacting entities (peers, software or human agents, things on the web, sensors and actuators) aiming to effectively share and search for information in an open and inherently distributed setting, need to agree on the meaning of the terms they use for structuring information, annotating data and conceptualizing the world. In any open setting, these entities can not be assumed to share the same ontology. However, each entity can only compute subjective correspondences of own ontology elements to the ontology elements of its acquaintances. Nevertheless, to propagate queries and enable effective sharing of information, the entities in the network need to reach an agreement on a globally coherent set of correspondences, wrt the semantics of specifications and their preferences on correspondences. Let us for instance consider two peers with ids 1 and 2, forming a cycle in a network. In this case, for each subjective equivalence correspondence from 1 to 2 for a pair of ontology elements (E_i^1, E_j^2) , peer 2 must agree that E_j^2 is equivalent to E_i^1 so that queries and results are propagated among them effectively.

This paper proposes a generic decentralized method for peers in a data management system to reach semantic agreements regarding ontology elements' subjective correspondences, via belief propagation. By "*generic*" we mean a method that is independent of the specific mapping methods used by the peers, and also applicable to any possible setting of information sharing entities. It must be noticed that although the proposed approach can be applied to any setting of interacting entities, for the clarity of the presentation we refer to peers in data

* AT2012, 15-16 October 2012, Dubrovnik, Croatia. Copyright held by the author(s).

management systems and to interacting agents that implement decision-making processes on behalf of these peers.

According to the proposed approach, each peer computes correspondences of its ontology elements to the elements of acquaintances' ontologies, using any set of mapping methods. To preserve the coherency of local mappings with respect to the semantics of specifications, peers build *internal* dependency graphs whose nodes (agents) are inter-constrained. These agents are responsible to compute specific mapping decisions on behalf of the peer: Each for a specific ontology element. Having computed subjective correspondences, peers aim to reach agreements to their mapping decisions: By exploiting the transitive closure of mappings in existing cycles in their network, peers can detect disagreements and revise their decisions.

The proposed method enables peers to revise their decisions with respect to their preferences, the semantics of specifications and the feedback they receive from others. The problem is addressed as a coordination problem where agents target the establishment of agreements wrt to the semantics of specifications, increasing the social welfare, i.e. the sum of the utilities of the individual agents. Toward this target we use the max-plus [8] algorithm to generate near-to-optimal solutions to reaching semantic agreements.

2 Problem Statement and Overall Approach

The problem of computing mappings (mapping relations or correspondences) between ontologies can be stated as follows: Given two ontologies $O_1 = (S_1, A_1)$, $O_2 = (S_2, A_2)$ (where S_i denotes the signature and A_i the set of axioms that specify the intended meaning of terms in S_i) and an element (class or property) E_i^1 in the signature S_1 of O_1 , locate a corresponding element E_j^2 in S_2 , such that a mapping relation (E_i^1, E_j^2, r) holds between them. r can be any relation such as the equivalence (\equiv) or the subsumption (\sqsubseteq) relation. For any such correspondence a mapping method may relate a value γ that represents the preference to relating E_i^1 with E_j^2 via r , i.e. the confidence on this correspondence. In this case the correspondence is denoted by $(E_i^1, E_j^2, r, \gamma)$. In this paper we deal only with *equivalences* between *classes*, although other mapping relations whose transitive closure can be exploited, can be considered as well. The set of mapping relations produces the mapping function $f : S_1 \rightarrow S_2$ that must preserve the semantics of representation: i.e. all models of axioms A_2 must be models of the translated A_1 : i.e. $A_2 \models f(A_1)$.

Let us consider a network of peers represented by a directed graph $G = (V, E)$. Each node in V is a peer P_{node} equipped with a specific ontology O_{node} . Each directed edge (P_{node}, P_{neigh}) in E specifies the fact that P_{node} has a specific view of P_{neigh} 's ontology (which can be either a complete, partial, vague or an abstract view, at the conceptual or at the assertional level, depending on peers' actual relation). P_{node} can propagate queries/ answers to P_{neigh} by computing correspondences between the elements of O_{node} and O_{neigh} . Generally, these are correspondences from the subjective point of view of P_{node} . In case there is also

an edge (P_{neigh}, P_{node}) , then the subjective correspondences from P_{neigh} to O_{node} might not be symmetric to those computed by P_{node} . This may happen even in the special case that these peers use the same mapping method (because the information that this method exploits may differ in each direction).

In such a setting, the problem of computing a coherent set of correspondences between peers in G is rather complicated if we consider that each peer is connected to numerous other peers with distinct ontologies and mapping methods, with the latter being arbitrarily connected with others and/or among themselves. This problem includes: (a) The computation of locally coherent correspondences to the ontologies of neighbor peers, and (b) the exploitation of the transitive closure of correspondences between ontologies to reach agreements between (even distant) peers¹. To describe our approach we distinguish between these, actually highly intertwined, computation phases.

Concerning the computation of locally coherent subjective correspondences of each peer P_{node} , given O_{node} (the ontology of P_{node}) we address the computation of correspondences from O_{node} to any ontology O_{neigh} of a neighbor peer P_{neigh} , as a coordination problem between self-interested agents. Each such tuple (O_{node}, O_{neigh}) is considered as a specific, distinct *mapping case* for P_{node} . Considering that P_{node} has at its disposal a mapping method A^{node} , for each mapping case there is a distinct set of agents, each of which corresponds to the mapping method and an ontology element in O_{node} , as it is also done in [19]: Each agent is responsible to decide on the mapping of that specific ontology element. We call these agents P_{node} – *internal* (or simply *internal*) agents to emphasize their role to compute correspondences on behalf of P_{node} . Actually, they may be distant and are organized in a certain way. Section 5.1 describes how the internal agents are related in dependency graphs (one distinct graph per mapping case) via validity constraints that must be satisfied so as the calculated mappings to conform to the semantics of ontological specifications.

Therefore, viewing the computation of ontology correspondences from O_{node} to O_{neigh} as a coordination problem between M ² P_{node} -internal agents that are related via validity constraints, we aim to find the joined state of agents (i.e. the overall set of correspondences), \mathbf{X}^* , such that the social welfare of the whole system (i.e. the sum of the individual agents' utilities U_m) is maximized:

$$\mathbf{X}^* = \arg \max_{\mathbf{X}} \sum_m U_m(x_m) \quad (1)$$

Given the subjective local decisions concerning correspondences, each of the peers propagate local mappings to the rest of the peers in the network in an iterative and cooperative manner, aiming to detect cycles of correspondences and get feedback on its decisions. Therefore, computed correspondences

¹ Notice that, even if correspondences are subjective, peers consider the correspondences computed by the others in conjunction to own correspondences, in order to exploit their transitive closure, trying to identify disagreements, get feedback and reach consensus.

² M in the general case depends on the number of elements in O_{node} and the number of alignment methods used by P_{node} .

($E_i^{node}, E_j^{neigh}, r, m$) for a mapping case (O_{node}, O_{neigh}) are sent to any of the immediate neighbors P_{neigh} that use O_{neigh} . In this case the parameter m does not denote the preference γ computed by the mapping method, but is an approximation of the payoff contribution related to that correspondence: This will be thoroughly explained in Section 5.2. P_{neigh} in its own turn dispatches received correspondences to the appropriate P_{neigh} -internal agents (i.e. to the agents that are responsible for computing a correspondence for E_j^{neigh} ³). Computed correspondences for P_{neigh} ($E_j^{neigh}, E_k^{xyz}, r, m$) are appended to ($E_i^{node}, E_j^{neigh}, r, m$) and are forwarded to P_{neigh} neighbors, and so on and so forth. The ordered list of correspondences forwarded to a peer shows the correspondences computed along a path and constitutes a *mapping history*. Although there are several strategies for propagating correspondences, in this work we assume that mapping histories are propagated in an unrestricted way, aiming to detect any cycle of correspondences in the network. Although we may further "tune" the efficiency of the propagation method, it must be pointed out that even with this naive propagation approach, the communication cost is not very high per ontology element, given that each peer propagates each message only to neighbor peers that correspond to a specific mapping case. As already said, each peer dispatches these messages to its internal agents. Having said that, we have to point out that the size of messages grow linearly with the length of the paths traversed in the network of peers.

Some correspondences might map an element from one ontology to a semantically irrelevant element in another ontology. Our goal is not to provide probabilistic guarantees on the correctness of a mapping operation as it is done in [5]: We aim towards driving peers to re-consider their local mappings with respect to their preferences, the semantics of specifications, also in accordance to the feedback they receive from other peers via cycles of mappings, so as to reach agreements. Towards this goal, also in accordance to [5], we take advantage of the cycles existing in the graph of peers. Given a cycle ($P_{node} \rightarrow P_{neigh} \rightarrow \dots \rightarrow P'_{neigh} \rightarrow P_{node}$) we consider that for each correspondence ($E_i^{node}, E_j^{neigh}, \equiv, m$) forwarded from P_{node} to P_{neigh} (together with the mapping history) the originator P_{node} must get a correspondence ($E_k^{neigh}, E_i^{node}, \equiv, m'$) from the last peer in the cycle, P_{neigh} . In other words, when P_{node} computes a local correspondence of E_i^{node} to E_j^{neigh} , then P_{neigh} , via the path from P_{neigh} to P_{node} must propagate a correspondence to the element E_i^{node} , rather than to any other element of O_{node} . In such a case, for each such correspondence, P_{node} counts a positive feedback. In case this does not happen, then there are one or more mappings through this path that according to the subjective view of P_{node} result to a disagreement. Then, P_{node} counts a negative feedback. It must be noticed that disagreements may still exist when P_{node} gets the expected correspondence but several correspondences along the path compensate their errors: These are detected by the corresponding peers as the mapping history propagates in the network.

³ There is one such agent in each dependency graph corresponding to a mapping case.

Summing up the above, each peer (a) gets a list of correspondences propagated to it (the mapping history), and (b) inspects the mapping history to detect a cycle. This is done by detecting in the mapping history the most recent correspondence originated from it. Such a correspondence concerns one of its own ontology classes and a specific mapping case. In case there is a cycle, the peer propagates the history to the internal agents that (i) correspond to that mapping case and, (ii) are responsible to compute correspondences for that class. These agents incorporate the feedbacks in their decision-making. In case there is not any cycle, histories are propagated to all internal agents corresponding to that mapping case. (c) After internal agents' computations, the peer propagates mapping decisions (together with the mapping histories) to neighbor peers corresponding to the mapping case. The propagation of beliefs in internal graphs and between peers, together with the incorporation of feedback to the algorithm are presented in detail in Section 5. It must be pointed out that belief propagation between peers happens seamlessly to the local computations of correspondences by internal agents. So, we may consider that a large, distributed graph comprising the internal agents of peers, spans the whole network of peers.

3 Related work

There are several works aiming to preserve the consistency of mapping decisions by locating (minimal) subsets of mappings that introduce inconsistencies [13],[14],[15],[17]. However, we consider distributed methods for the computation of agreed correspondences between ontologies of peers in a network.

Argumentation methods support the creation and exchange of arguments, followed by reasoning on their acceptability. There are numerous approaches that explore the application of argumentation methods for reaching semantic agreements between agents (e.g. [20],[21],[12]). Arguments are being used as positions for or against correspondences. These methods are proposed mostly for cases where peers need to produce mappings between their shared ontology and the ontology of a third party: I.e. to a rather restricted set of settings. Alternatively to arguments, we propose agents to exploit estimations of correspondences' payoffs, which are propagated to other agents and peers.

Probabilistic message passing techniques proposed in [5] aim to detect erroneous mappings between peers in a peer-to-peer data management system. This method targets cases where peers connected in a network have different ontologies and produce mappings to each other. The proposed method takes advantage of the transitive closure of mapping operations in a cycle of peers, to compare a query q to the returned q' via that cycle. To a certain extent than that work, we aim to compute correspondences that maximize the social welfare of the overall system (leading agents to agreement), with respect to the semantics of specifications and to the preferences of agents.

For the establishment or development of a common vocabulary among entities, researchers have proposed cognitive models [16], bilateral communication protocols and strategies [7][18] for agents, also studying the properties of shared

vocabularies [6], or the dynamics of their establishment [1]. In this paper we start from the point where peers have own ontologies (assumed to be developed/learned independently from the ontologies of others). We do not deal with how ontologies have been learned, or how effective mappings have been established between pairs of peers: This may be achieved either using an instance based method [7][22], a method for "tuning" the descriptions of concepts [22], or other ontology mapping methods. The proposed method aims at reconciling differences between computed correspondences, based on the notion of feedback, whose effect has also been studied in [16],[21],[5].

4 The max-plus algorithm

Before proceeding to the details of the computations, this section briefly explains how belief propagation happens via the max-plus algorithm.

One can deal with a complicated problem by viewing it as a factorization of several local functions, each depending on a subset of the variables that appear in the global problem. In particular, aiming to optimize over a function that is decomposed in local terms, we may compute the maximum a posteriori configuration in a graphical model using the max-product algorithm [11]. In our case, we aim at selecting the appropriate correspondences among ontology elements so as to maximize the social welfare of a set of agents. Message passing algorithms, and particularly the max-plus algorithm [8,10] are directly applicable to such a problem. In the most general case, considering a graph of interacting agents (a dependency graph - aka coordination graph [9]- specifying how the decisions of one agent affect the decisions of others), in order to compute the optimal joint state \mathbf{X}^* according to equation (1), each agent (node in the graph) sends a message μ_{ij} to each of its neighbors A_j in $N(A_i)$. This message can be regarded as a local payoff function [10] of agent A_j and it is defined as

$$\mu_{ij}(x_j) = \max_{x_i} \{U_{ij}(x_i, x_j) + \sum_{A_k \in N(A_i) - \{A_j\}} \mu_{ki}(x_i)\} + c_{ij} \quad (2)$$

where c_{ij} is a normalization factor. This message is computed by maximizing over the preferences of A_i , the sum of the utility U_{ij} and all incoming messages to A_i except from A_j . Messages are exchanged until they converge to a fixed point ⁴, or until an external signal is received (this is the case in this paper: The algorithm iterates I times). Since payoff functions are defined over at most two agents the utility U_{ij} can be of the form

$$U_{ij}(x_i, x_j) = f_i(x_i) + f_i^j(x_i, x_j) \quad (3)$$

where f_i specifies the payoff contribution of agent A_i , and f_i^j the payoff contribution of the pair of neighbor agents A_i and A_j . These functions are specified in Section 5.1.

⁴ Unfortunately, there are no guarantees that the max-plus converges to optimal solutions in graphs with cycles and no assurances can be given concerning the quality of the joint state \mathbf{X}^* .

The agent A_i may at any time step compute

$$g_i(x_i) = f_i(x_i) + \sum_{A_k \in N(A_i)} \mu_{ki}(x_i) \quad (4)$$

which equals the contribution to the global payoff function achieved via A_i 's neighborhood. Thus, each agent can form a decision by approximating its optimal choice by computing

$$x_i^* = \arg \max_{x_i} g_i(x_i) \quad (5)$$

Messages may be updated either asynchronously or synchronously. Any node in the graph may initially sent an identity (i.e. $\mu_{ij}(x_j) = 1$). When a node receives messages from its neighbors it computes outgoing messages.

5 Decentralized Coordination Through Belief Propagation

5.1 Internal Dependency Graphs

Considering one of the mapping cases of P_{node} , from O_{node} to O_{neigh} , P_{node} constructs a dependency graph for this mapping case on-the-fly: The nodes of this graph are P_{node} -internal agents. As already said, each agent A_i^{node} corresponds to an ontology class E_i^{node} of O_{node} , given a specific mapping method A of P_{node} .

The state of an internal agent A_i^{node} is represented by a tuple $(x_i^{node}, \gamma(x_i^{node}))$. The range of x_i^{node} is the set of classes in O_{neigh} that the method A assesses to be related to E_i^{node} via a relation r (in this paper equivalence only). Let that set be D_i^{node} . A value selection for x_i^{node} (i.e. a class in D_i^{node}) is denoted by $\text{value}(x_i^{node})$. The ordering imposed to D_i^{node} by γ represents the preference of the agent to each possible value of x_i^{node} . In case elements in D_i^{node} are not associated with confidence values, then the agent considers all possible values of x_i^{node} to be equally preferred. Such an agent A_i^{node} is depicted in Figure 1(a). Therefore, the internal dependency graph corresponding to the mapping case (O_{node}, O_{neigh}) with a single mapping method, comprises at most $M_{node} = |S_{node}|$ agents (S_{node} is the signature of O_{node}).

Each agent A_i^{node} interacts with a limited number of other agents: In the most general case, given a specific mapping case, the P_{node} -internal neighbors of A_i^{node} are those agents that correspond to ontology classes that are direct subsumees of E_i^{node} in O_{node} . Formally, the set of P_{node} -internal neighbors of an agent A_i^{node} is denoted by $N(A_i^{node})$ and it is defined to be the set

$$N(A_i^{node}) = \{A_n^{node} | (n \neq i) \text{ and } (E_i^{node} \sqsubseteq E_n^{node} \text{ in a direct way}) \}$$

Agents in P_{node} -internal graphs are interrelated with constraints that affect their states (i.e. mapping decisions). Let us first show this with an example (the corresponding dependency graph is shown in Figure 1): Given a mapping case

with two ontologies O_{node} and O_{neigh} and a mapping method A , the dependency graph comprises agents (some of them are depicted in Figure 1 in round-corner rectangular), related with the mapping method, a specific class from O_{node} , and a state variable x). The method A computes for each class E_i^{node} the set D_i^{node}

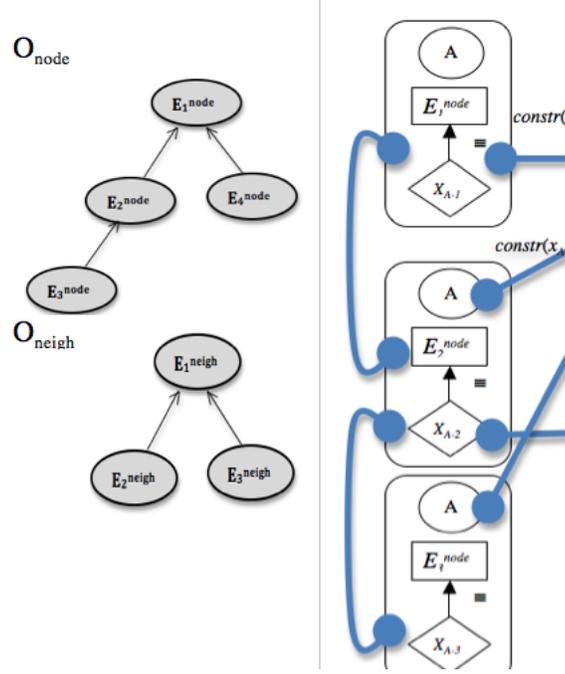


Fig. 1. Ontologies (Mapping Case) and Dependency Graph (part)

of candidate corresponding classes from the ontology O_{neigh} . Each of the classes in this set is assessed to be related to E_i^{node} via the equivalence relation, with a specific degree of confidence. Given two classes from O_{node} , s.t. $E_2^{node} \sqsubseteq E_1^{node}$, The agent for the class E_1^{node} considers correspondences to classes of O_{neigh} such that $value(x_1) \equiv E_1^{node}$. Similarly, the agent for the class E_2^{node} considers values - i.e. classes of O_{neigh} - for the variable x_2 such that $value(x_2) \equiv E_2^{node}$. Clearly, since $E_2^{node} \sqsubseteq E_1^{node}$, to maintain consistency between the mappings computed by these agents it must hold that $value(x_2) \sqsubseteq value(x_1)$. This is a validity constraint (shown with a bold line between agents) that must be preserved by the agents A_1^{node} and A_2^{node} and it is specified by the constraint $constr(x_2, x_1, \sqsubseteq)$.

Generally, the constraints between two agents A_i^{node} , A_u^{node} , are as follows (given that the mapping methods considered compute equivalences, only):

If $((E_j^{node} r E_w^{node})$ and $(value(x_w) \equiv E_w^{node}$ and $value(x_j) \equiv E_j^{node})$,
then $constr(x_j, x_w, r)$, where $r \in \{\equiv, \sqsubseteq\}$

Validity constraints specify how the utility of each agent is affected by its own state and the state of its neighbors. Recalling that the utility has the generic form specified in (3), payoff functions f_i and f_i^j are as follows:

$$f_i(x_i) = \gamma(x_i) \quad (6)$$

$$f_i^{k-l}(x_i, x_{k-l}) = \begin{cases} 1 & \text{if } \text{constr}(x_i, x_{k-l}, r) \text{ satisfied} \\ 0 & \text{in any other case} \end{cases} \quad (7)$$

where $\gamma(x_i)$ is the confidence by which the mapping method maps the class E_i^{node} to the class $value(x_i)$ of the other ontology. $N(A_i)$ is the set of A_i neighbors, and $\text{constr}(x_i, x_k, r)$ denotes the constraint that $value(x_i)$ must be related to $value(x_k)$ via r .

5.2 Belief Propagation Among Peers

Each edge (P_{node}, P_{neigh}) in the network of peers, corresponds to a distinct mapping case (O_{node}, O_{neigh}) for P_{node} . Each P_{node} -internal agent A_i^{node} for this mapping case ("responsible" to compute correspondences for the class E_i^{node} to O_{neigh}), computes the messages to its neighbor agents according to the formula (2). Doing so, the agent A_i^{node} communicates its payoff function to the agents with whom it shares a constraint. Nevertheless, to reach agreements, the state of each P_{node} -internal agent A_i^{node} must affect also the computations of (distant) peers in cycles with P_{node} , and visa-versa. Therefore, the states of P_{node} -internal agents must be propagated to neighbor peers. It must be pointed out that no peer knows any part of the internal graph or any of its neighbor peers. Therefore messages are propagated among peers, rather than among internal agents of different peers.

As already said, messages concerning correspondences for a mapping case are propagated to all neighbor peers corresponding to that mapping case. Doing so, peers may detect cycles, take the full advantage of the transitive nature of cyclic mappings, and detect all positive/negative feedbacks provided by agents. Specifically, given the edge (P_{node}, P_{neigh}) , P_{node} collects the states of each P_{node} -internal agent corresponding to the mapping case from P_{node} to P_{neigh} in a synchronous way (i.e. after I iterations of the max-plus algorithm) and propagates these decisions to P_{neigh} , and to all other peers corresponding to that mapping case. Actually, P_{node} propagates messages, each including a mapping history and a new correspondence of the form $(E_i^{node}, E_j^{neigh}, \equiv, m)$. E_j^{neigh} is computed according to equation (5). The actual form of m is $m_{node}^{neigh}(E_j^{neigh})$ and it is equal to $g_i(E_j^{neigh})$, given by formula (8) below, which refines the formula (4). The computations are performed by the internal agents of P_{node} that are responsible to compute correspondences for any element E_i^{node} .

P_{neigh} , by inspecting any mapping history that has been propagated to it, can detect the existence of a cycle, and can spot one of its internal agents that is interested to the propagated mappings: As already said, this is done by detecting in the history the most recent correspondence originated from it. This

correspondence (it exists) concerns one of O_{neigh} classes. The agent to which the feedback and mapping history is propagated is the one that decides on correspondences for that class and for that mapping case. In case there is not a cycle, the messages are propagated to all internal agents corresponding to that mapping case.

P_{neigh} -internal agents exploit incoming messages to detect any feedback and incorporate these into their decision. Specifically, after I iterations of the max-plus algorithm, P_{neigh} -internal agents compute own decisions by refining the computation of payoffs specified by formula (4), considering also feedback messages, as follows:

$$g_i(x_i) = f_i(x_i) + (\alpha \times \sum_{A_k \in N(A_i)} \mu_{ki}(x_i)) + (\beta \times \sum_{node} F_{node}(x_i)) \quad (8)$$

where $node$ denotes any P_{node} that has propagated correspondences to P_{neigh} . In case there is a closing cycle with a negative feedback, or in case there is not a closing cycle, $F_{node}(x_i)$ is 0, else $F_{node}(x_i)$ is $m_{node}^{neigh}(x_i)$. In other words, correspondences are decided by taking into account any feedback received from neighbor peers by exploiting mapping histories corresponding to the mapping case considered. Also, α and β are parameters (real numbers ranging in $[0,1]$) that have been used to experiment with the effect and necessity of feedback and constraints on agents' decision making.

6 Experimental Evaluation

We have conducted experiments with different networks, and with different peers' abilities to compute correspondences.

For each experimental setting (presented below) we measure (a) the recall R and the precision P of the correspondences computed by each peer for each mapping case (with respect to the reference mappings), (b) the number of positive and negative feedbacks F^+ and F^- received by each peer, (c) the number of messages PM received by each peer in each run, and (d) the number of messages AM exchanged between the internal agents of each peer per run.

To measure the degree / level of agreement achieved between peers and the effectiveness of computations we measure the f -score (specified by $f\text{-score} = 2 * (P * R) / (P + R)$) for each mapping case, the level of agreement achieved for each peer, (specified by $Level = (F^+ - F^-) / PM$), the agreement accuracy for pairs of peers, (specified by $Accuracy = Level * f\text{-score}$), and the *Message Gain* for achieving this accuracy (and level) given the number of inter-agent messages, specified by $Gain = (Accuracy * sf) / AM$, where sf is a scaling factor set equal to 1000. The motivation for these proposed scores is as follows: While f-score shows in a combined way the precision/recall of the correspondences computed per mapping case, the level of agreement shows (independently of the precision/recall achieved) the agreement achieved between peers as a function of positive and negative feedbacks after a specific number of inter-peer messages:

Peers may agree to false positive correspondences, as well. To provide a measure of these correspondences, the agreement accuracy measure combines the level of accuracy and the accuracy of computed mappings. Finally, aiming to show the efficiency of the method, we provide the gain as a function of the accuracy achieved and the number of inter-agent messages.

Given that the complexity of the method increases as the length and number of cycles in the network of peers increase (rather than with the number of peers), and in order to study the various patterns of method's behavior, we demonstrate the performance of the method with networks of $N=6$ peers, each with an increasing complexity. Specifically, experiments concern four types of networks in which peers' average degree is 2,3,4 or 5. Each type of network comprises at most 10 different networks. For each type of network we run experiments with various values of α and β , distinguishing the following cases:

- (C1) $\alpha \in (0, 1]$ and $\beta \in (0, 1]$: In these cases we need agents' decision to be affected by both, the feedback received, and by the inter-agents constraints, in a *balanced* way. In this case we expect agents to achieve high f-scores and high levels of agreements. Since it is not clear which case is a "balanced" case, we have run several experiments with α and β ranging from 0.1 to 0.9. Below we present experiments from the following combinations of α and β :
 - (case1) ($\alpha = 0.1, \beta = 0.9$),
 - (case2) ($\alpha = 0.3, \beta = 0.7$),
 - (case3) ($\alpha = 0.5, \beta = 0.5$),
 - (case4) ($\alpha = 0.7, \beta = 0.3$),
 - (case5) ($\alpha = 0.9, \beta = 0.1$).
- (C2) $\alpha = 1$ and $\beta = 0$: In this case agents' decision is affected by the inter-constraints, rather than by the feedback received. In this case we may expect agents to achieve high f-scores but low levels of agreement. However, experiments show that actually, constraints play a crucial role in achieving high levels of agreements.
- (C3) $\alpha = 0$ and $\beta = 1$: In this case agents' decisions are affected mostly by the feedback received, rather than by the inter-agents constraints. In this case we may expect high levels of agreement but low f-scores. Nevertheless, given that agents' decisions are not inter-constrained, feedback alone does not drive agents to reach high levels of agreement.
- (C4) $\alpha = 0$ and $\beta = 0$: In this case agents compute their decisions in a purely random way. Neither inter-agents' constraints nor feedback messages affect their decision. This is a baseline case where f-core and agreement measures are expected to be very low.

For each of the above cases we ran 10 independent experiments for each type of network (thus, 50 experiments for each case and sub-case, i.e. 400 experiments in total). Below we present aggregated results: The average of the scores achieved per case, and the average performance of one peer (over the different networks considered) in two cases.

In each of these cases each peer is been randomly assigned with a specific distinct ontology with 13 classes.

As far as the ability of peers to compute correspondences is concerned, in order to have a control on the experimental settings, each peer is set to compute mapping relations with a constant precision in $\{1,0.9,0.8,0.7,0.6,0.5\}$, in any of the networks considered. The precision specifies the proportion of the decided correspondences that are correct (true positives). False positive correspondences map a class to a randomly chosen class with high preference, while the correct correspondence for that class is assigned a random, low preference. As the difference between the mapping ability of neighbor peers grows, as the problem for reconciling different views and thus, computing agreements, is expected to become harder: As the average degree of peers increases, each peer has neighbors with quite different mapping abilities.

The max-plus running in each of the internal agent graphs iterates for a fixed number of iterations (I) per round. After experimentation I has been set to 5.

First, we have run experiments to determine what constitutes a "good" balance between constraints and feedback (i.e. to determine a good option for α and β for the case C1). Then, given the best option, we compared this case with the rest of the cases C2, C3, C4, where α and β range in $\{0,1\}$, excluding the case where α and β are both equal to 1.

Average results from the 5 sub-cases of case C1 are shown in Figure ??(a).

As Figure 2(a) shows, the performance scores, and especially the f-score and the level of agreement achieved in all cases, are nearly the same: In all cases agents compute their decisions wrt their constraints with other internal agents and to the feedback received from other peers. However, case 2 ($\alpha = 0.3, \beta = 0.7$) achieves a slightly higher f-score (0.86) and level of agreement (0.10) than the other cases. Therefore, we consider this to be the "balanced" case. Subsequently, when we refer to C1 we refer to this particular case. Having said this, we must point out that the method is robust to the choice of the external parameters α, β , although a "good" choice can make a slight difference to the results. Figure 2(b) shows the performance measures for a specific (representative) peer for the C1 for each of the rounds of the overall method (measurements are the averages for 10 independent experiments). As we may notice, the proposed method (in all the cases and for each of the peers) manages to gradually increase the f-score and the number of positive feedback received, thus the level of agreement between peers. Also, the method manages to reach a plateau after round 5.

Having determined the "balanced" C1 case, we can compare the results achieved in that case, with the average f-score, agreement levels, accuracy and message gain achieved in all cases where α and β range in $\{0,1\}$. Figure 2(c) shows the results for all the cases (C1 to C4). As expected, the best case is C1 (i.e. the balanced case), followed by C2 with f-score 0.85 and level of agreement 0.09. It may be a surprise that C2 achieves a high level of agreement, but this is due to the fact that all peers have the same ontology. In this case, peers' internal agents form mapping decisions that respect the semantic constraints, which are shared by all groups of internal agents in each peer. This results to a high level of agreement even if feedback is not considered by the agents. Although this is the case for structurally similar ontologies, future work aims to investigate

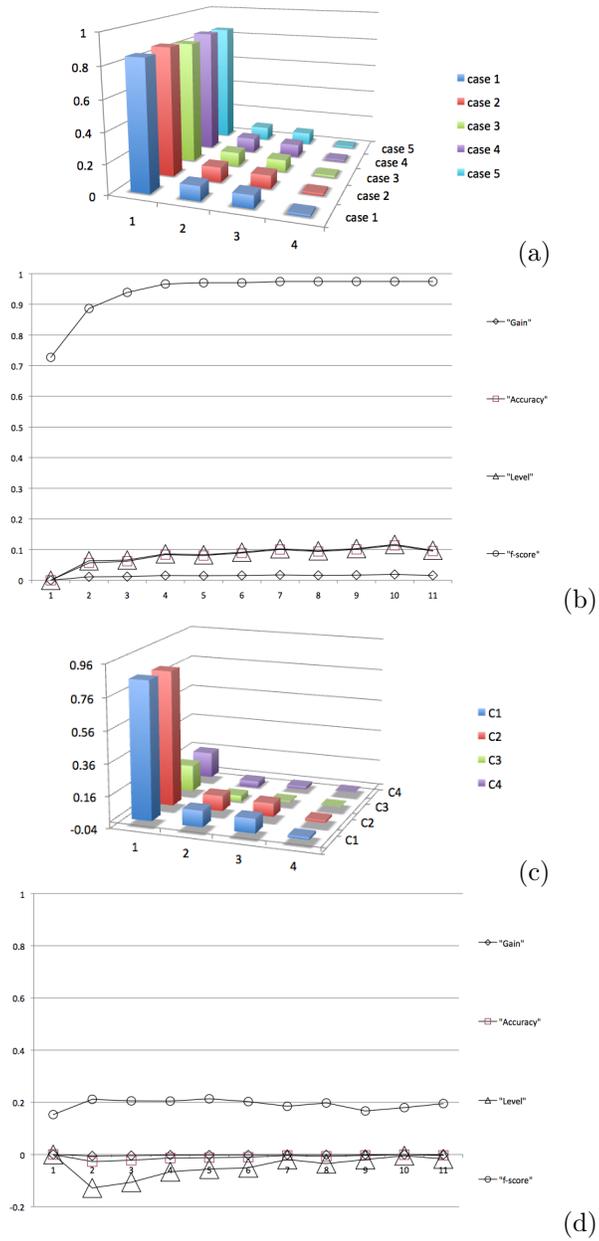


Fig. 2. Experimental Results

this case more intensively, considering ontologies that can be aligned but that are structurally dissimilar. For the rest of the cases, i.e. C3 and C4, both the

f-score and the level of agreement achieved are very low. However, in C3 the level of agreement is slightly higher than C4. In both of these cases, the fact that constraints are not being considered, causes the unconstrained fluctuation of agents' decisions among all possible correspondences, thus affecting the level of agreement achieved. Figure 2(d) shows the (representative) performance of a specific peer in case C3. Again, the reported measurements are the averages for 10 independent experiments.

Concluding the above, experimental results show that semantic coordination must consider both, the constraints and the feedback received, although semantic constraints play a vital role in the process.

7 Concluding Remarks and Future Work

We address the semantic coordination problem in an agent-based data management system through payoff propagation, by a distributed extension of the max-plus algorithm. The method is shown to be effective enough, making the best of the mappings produced by peers, helping peers increase their levels of agreement and the agreement accuracy. Future work concerns including other mapping relations whose transitive closure can be exploited by peers (e.g. subsumption and disjointness) in cycles. Also, we aim to address subtle issues concerning communication; although - via experiments performed- various heuristics for reducing the communication messages result to sacrificing considerably methods' effectiveness. Finally, the study of sub-optimality, as well as the integration of the proposed method to a peer-to-peer setting seamlessly to other distributed processes (e.g. information retrieval) are important issues.

References

1. Baronchelli A., Felici M., Caglioti E., Loreto V., L. Steels 2006 Sharp Transition towards Shared Vocabularies in Multi-Agent Systems. *J. Stat. Mech.* P06014, 2006.
2. Beun R-J, van Eijk M. R., H. Prust, 2004 Ontological Feedback in Multi-Agent Systems. In *Proc. of AAMAS 2004*.
3. Cudre-Mauroux P., et al. 2006 Viewpoints on emergent semantics. S. Spaccapietra et al. (Eds.): *J. on Data Semantics VI*, LNCS 4090, pp. 1-27, 2006.
4. Cudre-Mauroux P., et al. 2009 idMesh: Graph-Based Disambiguation of Linked Data. In *Proc of WWW 2009*.
5. Cudre-Mauroux P., Aberer K., and A. Feher 2006 Probabilistic Message Passing in Peer Data Management Systems, In *Proc. of ICDE 2006*.
6. Diggelen J., et al., 2005 Optimal Communication Vocabularies and Heterogeneous Ontologies. *Agent Communication (Rogier M. van Eijk et al eds.)*, LNCS 3396, 2005.
7. Diggelen J., et al. 2006 ANEMONE: An effective minimal ontology negotiation environment, In *Proc. of AAMAS 2006*.
8. Farinelli, A., Rogers A., Petcu A. and N.R. Jennings 2008 Decentralised coordination of low-power embedded devices using the max-plus algorithm. In *Proc. of AAMAS 2008*.
9. Guestrin, C., Koller D., Parr R. 2002 Multiagent planning with factored MDPs. *NIPS 14*, MIT Press 2002.

10. Kok J.R. and N.Vlassis 2006 Collaborative Reinforcement Learning by Payoff Propagation, *JMLR*, 7:1789-1828, 2006.
11. Kschischang, F.R., Frey B.J., Loeliger H-A., 2001 Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, Vol 47, Issue 2, 2001.
12. Laera L., et al. 2007. Argumentation over ontology correspondences in mas. In *Proc. of AAMAS 2007*.
13. Meilicke C. and H. Stuckenschmidt 2009 An efficient method for computing alignment diagnoses. In *Proc. of RR 2009*.
14. Meilicke C., Stuckenschmidt H. and A. Tamilin 2009 Reasoning support for mapping revision. *Journal of logic and Computation*, 19(5):807-829, 2009.
15. Qi G., Q. Ji, and P. Haase 2009 A conflict-based operator for mapping revision, in *Proc. of ISWC 2009*.
16. Reitter D. and C.Lebiere, 2011 How groups develop a specialized domain vocabulary: A cognitive multi-agent model. *Cognitive Sys. Res.* 12, , pp 175-185, 2011.
17. Ruiz E.J., Grau B.C., Horrocks I. and R.Berlanga 2009 Ontology integration using mappings: Towards getting the right logical consequences. In *Proc. of ESWC 2009*.
18. Sensoy M., P. Yolum, 2009. Evolving service semantics cooperatively: a consumer-driven approach. *J. of Aut. Agents and Multi-Agent Sys*, 18, 2009.
19. Spiliopoulos, V, Vouros G. In Press. Synthesizing Ontology Alignment Methods Using the Max-Sum Algorithm. *IEEE TKDE*, Vol 24, Issue 5, pp 940-951, 2012.
20. Trojahn C., Euzenat J. 2010 Consistency-driven Argumentation for Alignment Agreement. *Proc. of OM*, 2010.
21. Trojahn C., Moraes M., Quaresma P., and R. Vieir 2008 A Cooperative Approach for Composite Ontology Mapping, *J. on Data Semantics X*, LNCS 4900, pp. 237-263, 2008.
22. Williams B. A. 2004 Learning to Share Meaning in a Multi-Agent Systems. *J. of Aut. Agents and Multi-Agent Sys*, 8, 2004.
23. M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the 2nd Int'l Conf. on Multiagent Systems*, pages 401-408, 1996.