Synthesizing Ontology Alignment Methods Using the Max-Sum Algorithm

Vassilis Spiliopoulos and George A. Vouros

Abstract—This paper addresses the problem of synthesizing ontology alignment methods by maximizing the social welfare within a group of interacting agents: Specifically, each agent is responsible for computing mappings concerning a specific ontology element, using a specific alignment method. Each agent interacts with other agents with whom it shares constraints concerning the validity of the mappings it computes. Interacting agents form a bipartite factor graph, composed of variable and function nodes, representing alignment decisions and utilities, respectively. Agents need to reach an agreement to the mapping of the ontology elements consistently to the semantics of specifications with respect to their mapping preferences. Addressing the synthesis problem in such a way allows us to use an extension of the max-sum algorithm to generate near-to-optimal solutions to the alignment of ontologies through local decentralized message passing. We show the potential of such an approach by synthesizing a number of alignment methods, studying their performance in the OAEI benchmark series.

Index Terms—Coherence and coordination, constraint satisfaction, interoperability, ontologies.

1 Introduction

The problem of synthesizing different ontology alignment methods is vital considering that there is no such "best" method: Methods exploit different types of information (lexical, syntactic, structural, semantic), exploit different facets of the same type of information, consult (or do not consult) external resources, target to different types of mappings between ontology elements, and so on. Therefore, the synthesis of alignment methods that exploit different types of information and that compute different types of relations between elements (e.g., equivalence or subsumption relations) can be of great benefit.

However, there are important questions yet to be answered: What constitutes a "synthesis," what does it means to get an "optimal" synthesis (i.e., what does it mean to make the best of the results of the constituent/individual methods), and what are the exact requirements of such a synthetic task, are issues that have not been examined meticulously.

We may distinguish three basic types of synthesis of ontology alignment methods.

The "simple synthesis" or mere "composition of results": This is the most simple case where the results of individual methods are combined by specific operators, e.g., by taking the union of results [1], intersection of results, by selecting the most "preferable" results from each individual method, or by combining the methods' different confidence values with weighing schemas [2], [3], [4]. If results are provided with confidence values (e.g., probabilities or similarities), which is not always the case, these numeric results may need to be normalized so as to be comparable. While each combination type may be of benefit to the precision or recall of the resultant method, it is very difficult to assure the consistency of results with respect to the semantics of specifications, and there is no assurance that results are with respect to the preferences of the different methods in the best way. In other words, "optimality" of the combination is very difficult to be assured.

The "interactive synthesis" or "biased synthesis": This is a more elaborated type of synthesis where the results of a method are being provided as input to other methods [5], [6], [7]. Each method exploits the provided results according to its own "point of view," considering them in addition to the information provided by the source ontologies, producing new/updated results. These new results may in turn be provided to other methods or to the initial methods in a recursive way, and so on, until the synthesized whole converges. Beyond the fact that it is very difficult to synthesize more than two methods using this type of synthesis (e.g., one method has to consistently exploit the results of the other methods), there is no generic or systematic way to formulate such a synthesis, which heavily depends on the constituent methods. Furthermore, as in the "simple synthesis" case, there is no assurance that results are with respect to the preferences of the different methods in the best way.

The "model-based synthesis": This is an "in-first-principles" approach where individual methods provide results to a generic model that assures the consistency of the final results by integrating the preferences of the constituent methods with respect to the semantics of specifications. This approach has several advantages: 1) The results produced by each individual method are not biased by the results produced by the other methods, although they are affected by them, 2) one may construct different models for different

- *V. Spiliopoulos is with the AI Laboratory, Information and Communication Systems Engineering Department, University of the Aegean, Konitsis 11, Drosia Attikis 14572, Samos, Greece. E-mail: vspiliop@aegean.gr.*
- *G.A. Vouros is with the Department of Digital Systems, University of Piraeus, Greece. E-mail: georgev@unipi.gr.*

Manuscript received 25 June 2009; revised 12 July 2010; accepted 10 Dec. 2010; published online 7 Feb. 2011.
Recommended for acceptance by Y. Chen.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-06-0520.
Digital Object Identifier no. 10.1109/TKDE.2011.42.

types of methods (string based, resource based, structure based, logic based, etc.) and 3) optimality of the results with respect to the preferences of the individual methods and the semantics of specifications can be addressed.

This paper addresses the problem of synthesizing ontology alignment methods using a model-based synthesis approach. Specifically, it addresses this problem by maximizing the social welfare within a group of interacting agents (i.e., maximizing the sum of utilities of individual agents). Each agent is responsible for making an alignment decision concerning a specific ontology element, using a specific alignment method. Agents need to reach an agreement to the mapping of the ontology elements, consistently to the semantics of specifications and according to their preferences (which are being provided by the alignment methods and are reflected on agents' utility function). Interacting agents form a bipartite factor graph (FG), composed of variable and function nodes (representing alignment decisions and utilities, respectively). Such a representation allows us to use an extension of the max-sum algorithm to generate solutions to the alignment of ontologies via the synthesis of individual methods. This problem is being treated as an optimization problem through local decentralized message passing. We study the potential and the benefits of this approach by synthesizing a set of methods that are applied to the OAEI benchmark series [8].

The proposed synthesis approach aims to address the following requirements from a synthesis method:

- *Generality.* The proposed method aims to address the problem as a *generic* coordination problem among a number of agents, each having its own state and utility. The proposed method should be used to synthesize any set of mapping methods.
- *Optimality.* The proposed solution aims to maximize the social welfare of agents, i.e., the sum of their utilities, taking into account the preferences of individual agents (the preferences of the individual alignment methods) and the semantics of specifications. This means that agents (each corresponding to a specific ontology element and a specific mapping method) aim to *jointly* compute a coherent and consistent set of mappings that they consider are highly possible to be correct.
- *Scalability.* The proposed model may enforce a decentralized solution, where each agent has knowledge of and can directly communicate with a small number of neighbors, that is, a small proportion of the total number of agents. Doing so, the complexity of the calculations that each agent performs depends only on the number of its neighbors.
- *Expandability.* New individual methods must easily be added to the synthesis model, independently to the other synthesized methods. However, in our case the complexity of the computations that each agent performs is exponential to the number of synthesized methods.
- *Tolerance to failures of individual methods.* Even if one or more of the synthesized individual methods fail to compute mappings, then the synthesis must produce results as effectively, as the "best" (i.e., the more effective) synthesized method.
- *Provision of feedback to the individual methods.* The proposed synthesis approach must exploit the mappings computed by all individual methods to provide feedback for the validity of the mappings produced by each individual method. This aims to drive individual methods to improve their effectiveness by choosing mappings that are mostly preferred by them, in conjunction to being consistent with the mappings computed by the other methods.
- *Being a semantics-based approach.* As it has already been mentioned, the proposed mappings must form a consistent set of mappings with respect to the semantics of ontology elements' specifications. To do so, the synthesis method must drive individual mapping methods to compute mappings that respect a set of validity constraints, subject to their preferences.

Toward addressing the above requirements, this paper makes the following contributions:

1. It formulates the synthesis of ontology alignment methods as a generic social welfare maximization problem. Doing so, it shows that the representation of this problem as a bipartite factor graph allows us to use an extension of the max-sum algorithm to generate solutions to the alignment of ontologies, which is being treated as an optimization problem through local decentralized message passing.
2. It thoroughly reports on the potential of the proposed approach to address the issues mentioned above by synthesizing three methods: The Classification-Based Learning of Subsumption Relations for the Alignment of Ontologies (CSR) method that, in contrast to the other two methods, computes subsumption relations among ontology elements, and the Vector-Space Model (VSM) and COCLU methods that compute equivalence relations among ontology elements. Also, each method is different from the others in the type of information that it exploits.

2 Problem Statement

An ontology is a pair $O = (S, A)$, where S is the ontological signature describing the vocabulary (i.e., the terms that lexicalize ontology elements) and A is a set of ontological axioms, restricting the intended meaning of the terms included in the signature. Considering two ontologies $O_1 = (S_1, A_1)$ and $O_2 = (S_2, A_2)$, and different types of relations between ontology elements, the ontology mapping problem can be stated as follows: Given *any* pair $(\mathtt{E}^1, \mathtt{E}^2)$ of elements of the input ontologies, such that $\mathtt{E}^{\mathrm{i}}$ is a term in $S_i, i = 1, 2$, *decide* on the exact relation (here, equivalence ($\equiv$) or subsumption (inclusion) ($\sqsubseteq$)) that relates these elements. Such a relation is denoted as a triple $(\mathtt{E}^1, \mathtt{E}^2, \mathtt{r})$, where $\mathrm{r} \in \{\equiv, \sqsubseteq\}$. Given this set of new relations, denoted by A_m, ontologies O_1 and O_2 can be aligned, resulting to a new consistent and coherent ontology $O = (S, A)$, such that $S = (S_1 \cup S_2)_{|\mu}$($\mu$ is a substitution list for the consistent

renaming of some terms), $A_m \subseteq A, A \models A_{1|\mu}$ and $A \models A_{2|\mu}.A_{i|\mu}$ denotes the translation of axioms in A_i according to the renaming of terms imposed by μ.

For any element $\mathtt{E}^2$ assessed to be related to $\mathtt{E}^1$ via a mapping relation $(\mathtt{E}^1, \mathtt{E}^2, \mathtt{r})$, an alignment method may relate a confidence value γ that represents its assessment/preference on relating $\mathtt{E}^1$ with $\mathtt{E}^2$ via $\mathtt{r}$. If there is not such a preference, we assume that the method equally prefers any such assessed relation for the element $\mathtt{E}^1$.

Given a set of K alignment methods, each method has its own preference concerning any assessed relation $(\mathtt{E}^1, \mathtt{E}^2, \mathtt{r})$. The synthesis of these K methods aims to compute an alignment of the input ontologies, with respect to the preferences of the individual methods. This is a coordination problem between the individual alignment methods.

We restrict the problem to the alignment of classes and we model it by considering a set of M interacting agents.

Each agent is *associated* with a specific alignment method AM_j and has a state that is described by a discrete variable x_{j-k} *corresponding* to a class $\mathrm{E}^{\mathrm{i}}_{\mathrm{k}}$ in one of the two input ontologies ($i = 1, 2$). The variable x_{j-k} ranges to the classes of the other input ontology that AM_j assesses to be related to $\mathrm{E}^{\mathrm{i}}_{\mathrm{k}}$ via a relation r.

Each agent interacts locally with a number of neighboring agents, such that its utility $U_i(\mathbf{X}_i)$ is dependent of its own state and the states of these agents (defined by the set $\mathbf{X}_i$). The form of this utility function reflects the mapping preferences of each agent with respect to the semantics of specifications, and it may not be known to other agents. The neighbors of an agent are determined by a set of validity constraints that must be satisfied so as the calculated mappings to conform to the semantics of specifications. The exact constraints, the form of agents' utility function and the set of agents' neighbors are specified in Section 4.

Viewing the synthesis as a coordination problem, we use the generic coordination method proposed in [9] to find the state of each agent, $\mathbf{X}^*$, such that the social welfare of the whole system (i.e., the sum of the individual agents' utilities) is maximized

$$X^* = \arg\max_X \sum_{m=1}^{M} U_m(x_m).$$

Before proceeding to the details of the proposed approach, we must clarify that the proposed model addresses the synthesis as a coordination problem between agents: The model comprises agents that correspond ontology entities and alignment mechanisms, and the network topology is determined by interagent validity constraints. The problem of synthesizing alignment methods using such a model is different (although closely related) from the one where agents exist in a specific network topology, have different ontologies, local utilities and preferences, and need to align their ontologies consistently by coordinating their computations of mappings: This problem is part of our future work and it is further discussed in Section 6 of this paper.

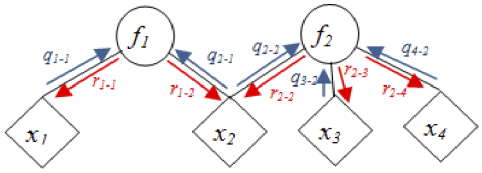


Fig. 1. Factor graph for the function g(X) = f1(x1, x2) f2(x2, x3, x4).

3 Factor Graphs for Decentralized Coordination

3.1 Brief Introduction to Factor Graphs and Message Passing Schemes

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. Let us for instance consider the function

$$g(\mathbf{X}) = f_1(x_1, x_2) f_2(x_2, x_3, x_4).$$

This factorization of g to the two local functions f_1 and f_2 can be represented by the bipartite factor graph depicted in Fig. 1, where variables (rectangles) are linked to their functions (ovals).

As it is shown in [10], factor graphs can be applied in many settings. In most of the cases, one is interested in computing a marginal function $g_n(x_n)$ of the global function (e.g., the marginal probability of a variable) that describes the total dependency of the function $g(\mathbf{X})$ on variable x_n

$$g_n(x_n) = \sum_{\{x_m\}, m \neq n} g(\mathrm{X}).$$

For instance

$$\mathrm{g}_2(\mathrm{x}_2) = \sum_{x_1} \sum_{x_3} \sum_{x_4} g(x_1, x_2, x_3, x_4) = \sum_{\{x_m\}, m \neq 2} g(\mathrm{X}).$$

Thus, considering our example, the marginal function g_2 can be rewritten as a series of sum-product operations on the local functions

$$g_2(x_2) = \sum_{x_1} f_1(x_1, x_2) \sum_{x_3} \sum_{x_4} f_2(x_2, x_3, x_4).$$

Marginals of all the variables can be computed simultaneously and efficiently by the sum-product algorithm. The algorithm propagates messages (depicted in Fig. 1 with labeled arrows) along the edges of the factor graph. To compute these marginals, the algorithm iteratively propagates messages of two forms:

1. Messages from variables to functions

$$q_{\mathrm{n}\to\mathrm{m}}(x_n) = \prod_{k \in M(n) \setminus m} r_{k \to n}(x_n). \quad (1)$$

2. Messages from functions to variables

$$r_{m \to n}(x_n) = \sum_{\mathrm{X}_m \setminus n} \left(f_m(\mathrm{X}_m) \prod_{l \in N(m) \setminus n} q_{l \to m}(x_l) \right), \quad (2)$$

where $N(m)$ is the set of variables connected to the function f_m, $M(n)$ is the set of functions connected to the variable x_n and $X_{m\backslash n} \equiv \{x_l : l \in N(m) \setminus n\}$.

For cycle-free graphs, the sum-product algorithm calculates the exact marginal functions for all variables, converging after a number of steps that is equal to the diameter of the graph [10]. Messages may be updated in any sequence (i.e., asynchronously). Leaf nodes in the graph may initially sent an identity (i.e., $q_{n\to m}(x_n) = 1$) message and function nodes a message $r_{m\to n}(x_n) = f_m(x_n)$. When a node receives a message from an edge, it computes outgoing messages according to (1) and (2) and sends the new messages via the remaining edges. When a variable receives all messages from its neighbors, it can compute the exact marginal function as the product of messages received

$$g_n(x_n) = \prod_{k\in M(n)} r_{k\to n}(x_n).$$

Considering our example, the computation of g_2 can be seen as the product of two messages $r_{1\to 2}(x_2)$ and $r_{2\to 2}(x_2)$ (shown in Fig. 1) sent, respectively, by f_1 and f_2 to x_2.

Rather than calculating the marginal functions of any variable, we may solve a global maximization task by calculating $\mathrm{argmax}_{\mathbf{X}}\, g(\mathbf{X})$ [9]. This can be done by replacing the summation operation in the "function to variables" messages by a max operator as follows:

$$r_{m\to n}(x_n) = \max_{\mathrm{X}_{m\backslash n}} \left(f_m(\mathrm{X}_m) \prod_{l\in N(m)\backslash n} q_{l\to m}(x_l) \right).$$

The product of messages flowing into any variable now represents

$$g_n(x_n) = \max_{\mathrm{X}_{m\backslash n}} \left(\prod_{m=1}^{M} f_m(\mathrm{X}_m) \right) \text{ such that if}$$

$$\mathbf{X}^* = \operatorname*{argmax}_{\mathrm{X}} \left(\prod_{m=1}^{M} f_m(\mathrm{X}_m) \right),$$

then each individual component is $x_n^* = \arg\max_{x_n} g_n(x_n)$.

For graphs that contain cycles, this maximization task is not guaranteed to converge and the results are suboptimal. For acyclic graphs, the results are optimal.

3.2 The Max-Sum Decentralized Coordination Algorithm

To synthesize ontology alignment methods, we aim at solving a social welfare maximization problem through local message passing. To do so, we use the max-sum decentralized algorithm introduced in [9]. In this section, we present the problem and the algorithm, while in Section 4 we present the formulation of the ontology alignment methods' synthesis by means of agents in a factor graph.

Considering a set of agents, we can represent each agent as a node that comprises a function (its utility) and a variable (its state). The utility of each agent is a function of its own state and of the state of agents that affect its own decision (i.e., its neighbors). Since in this case, we are aiming to maximizing a summation (the social welfare, i.e., the sum of agents' utilities) we operate in the logarithmic space, using the max-sum algorithm. In this case, the messages are as follows [9]:

From variable to function,

$$Q_{\mathrm{n}\to\mathrm{m}}(x_n) = \log q_{\mathrm{n}\to\mathrm{m}}(x_n) = a_{nm} + \sum_{k\in M(n)\backslash m} R_{k\to n}(x_n),$$

where a_{nm} is a scalar such that $\sum_{x_n} Q_{n\to m}(x_n) = 0$.

From function to variable,

$$\begin{aligned} R_{m\to n}(x_n) &= \log r_{m\to n}(x_n) \\ &= \max_{\mathrm{X}_{m\backslash n}} \left(U_m(\mathrm{X}_m) + \sum_{l\in N(m)\backslash n} Q_{l\to m}(x_l) \right). \end{aligned} \tag{3}$$

The marginal function of each variable is the sum of messages to this variable

$$G_{\mathrm{n}}(x_n) = \sum_{k\in M(n)} R_{k\to n}(x_n).$$

Since $\log g(\mathrm{X}) = \sum_{\mathrm{m}} \log f_m(\mathrm{X}) = \sum_{\mathrm{m}} U_m(\mathrm{X}_m)$, these marginal functions represent solutions to the social welfare maximization problem. In case the factor graphs contain cycles, the max-sum algorithm will approximate the marginal functions.

Each agent computes $\arg\max_{x_n} G_n(x_n)$ determining via local criteria, which state it should adopt so as to maximize social welfare. Furthermore, agents initialize their outgoing messages randomly, updating them whenever they receive an updated message from a neighbor.

4 Ontology Alignment Methods and their Synthesis

As already pointed in Section 2, given a set of K alignment methods AM_i, $i = 1 \ldots K$, we address the synthesis of these methods as a coordination problem between self-interested agents. To do so, we assume that each agent has knowledge of, and can directly communicate with few neighboring agents on whose state its own utility depends. This section describes how the network of acquaintances (i.e., the factor graph) is being constructed, the validity constraints among agents' states, as well as the exact form of agents' utility function.

Specifically, given the K alignment methods and two ontologies $O_1 = (S_1, A_1), O_2 = (S_2, A_2)$, the situation is represented by a factor graph that is being constructed on-the-fly: The nodes of this factor graph are the utilities (functions) and the states (variables) of agents. Each agent A_{j-i} corresponds to an ontology class $\mathtt{E}_{\mathrm{i}}^1$ of O_1 and to a specific alignment method AM_j. Without any loss of generality, we assume that agents correspond to the classes in the ontology O_1, which is called the "*base ontology*" and is being chosen randomly among the two input ontologies. The state of A_{j-i} is represented by a variable x_{j-i} that ranges in the subset of classes in O_2 that the method AM_j assesses to be related to $\mathtt{E}_{\mathrm{i}}^1$ via a relation r. Let that set be D_{j-i}. For the purposes of this paper, r can be either the equivalence ($\equiv$) or the subsumption (inclusion) ($\sqsubseteq$) relation. The possible values $\mathtt{E}_{\mathrm{i}}^2$ of x_{j-i} may have been assigned confidence values by AM_j. The ordering imposed to D_{j-i} by

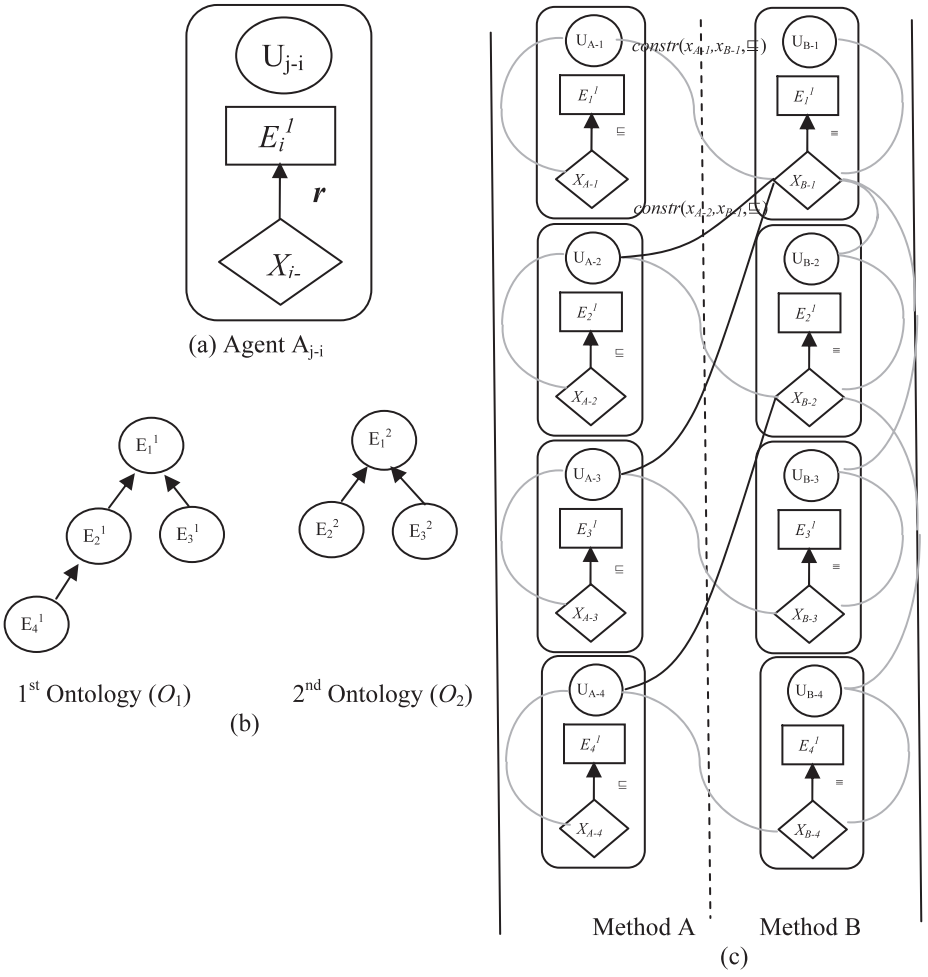


Fig. 2. (a) The representation of agents, (b) two ontologies, and (c) the corresponding factor graph.

the confidence values represents the preference of the agent to each possible value of x_{j-i}. In case that classes in D_{j-i} are not associated with confidence values, then the agent considers all possible values of x_{j-i} to be equally preferred. Each agent A_{j-i} (corresponding to the method AM_j and to the element $\mathrm{E_i}$ of the base ontology) is depicted as in Fig. 2a. Therefore, the factor graph comprises at most $|S_1| \times K$ agents, i.e., at most $2 \times |S_1| \times K$ nodes, in case O_1 is the base ontology. We should point out that if AM_j assesses that there are no classes that can be related to the class E_i^1, i.e., $D_{j-i} = \emptyset$, then no agent is created for the specific class E_i^1 and mapping method AM_j.

Each agent A_{j-i} interacts with a limited number of neighbors: The neighbors of A_{j-i} are those agents that correspond to classes that are direct subsumees of E_i^1 in the base ontology, and the agents that correspond to the class E_i^1 but are associated with different alignment methods or different type of alignment relations (r). Formally, the set of neighbors of an agent A_{j-i} is denoted by $\mathcal{N}(A_{j-i})$ and it is defined to be

$$\mathcal{N}(A_{j-i}) = \{A_{u-n} | [n = i \; and \; j \neq u] \; or \; [(n \neq i) \; and \; (\mathrm{E}_\mathrm{i}^1 \sqsubseteq \mathrm{E}_\mathrm{n}^1 \; in \; a \; direct \; way)]\}.$$

In this definition, without loss of generality, we assume that the base ontology is the first ontology.

The states of neighboring agents are interrelated with constraints that affect the values they may take. Let us first show this with an example. Given the two ontologies depicted in Fig. 2b (let O_1 be the base ontology) and two alignment methods A and B, the factor graph is as shown in Fig. 2c. Given that, as it is also shown in the graph, the method A computes only subsumption relations and the method B computes only equivalences between the two ontologies, the graph comprises eight agents, each for a method-class combination and a specific type of alignment relation. Specifically, we consider the generic case where the method A (respectively, B) computes for each class E_i^1 of the base ontology the set D_{A-i} (respectively, D_{B-i}) of candidate classes from the ontology O_2. Each of the classes in this set is assessed to be related to E_i^1 via the subsumption (respectively) relation. As already said, the computed sets of classes are sets of possible values for the variables of the corresponding agents: e.g., the first agent for the method A considers values—classes from the second ontology—for the variable x_{A-1} such that $x_{A-1} \sqsubseteq \mathrm{E}_1^1$. Similarly, the first agent for the method B considers values—i.e., classes from the second ontology—for the variable x_{B-1} such that $x_{B-1} \equiv \mathrm{E}_1^1$. Clearly, to maintain consistency between their mappings, it must hold that $x_{A-1} \sqsubseteq x_{B-1}$. This is a constraint that must be preserved by the agent A_{A-1} and it is specified by $constr(x_{A-1}, x_{B-1}, \sqsubseteq)$. This is depicted in

Fig. 2c by the curved line that connects the utility U_{A-1} to the variable x_{B-1}. Similarly, since $\mathrm{E}_2^1 \sqsubseteq \mathrm{E}_1^1$, it must hold that $x_{A-2} \sqsubseteq x_{B-1}$. This is specified by the constraint $constr(x_{A-2}, x_{B-1}, \sqsubseteq)$ for the agent A_{A-2} in Fig. 2c.

All the constraints are depicted in Fig. 2c by curved lines: In case, two variables x_{j-i}, x_{u-w} are associated with a specific constraint, then the utility of the agent A_{j-i} is associated with x_{u-w}.

Generally, the constraints between the variables x_{i-j}, x_{u-w} of two agents A_{i-j}, A_{u-w}, are generated as follows (given that the base ontology is the first input ontology):

- If $(\mathrm{E}_\mathrm{j}^1 \sqsubseteq \mathrm{E}_\mathrm{w}^1$ and $(x_{u-w} \equiv \mathrm{E}_\mathrm{w}^1$ and $(x_{i-j} \equiv \mathrm{E}_\mathrm{j}^1$ or $x_{i-j} \sqsubseteq \mathrm{E}_\mathrm{j}^1)))$ then $constr(x_{i-j}, x_{u-w}, \sqsubseteq)$. An example of such a constraint is the constraint $constr(x_{A-2}, x_{B-1}, \sqsubseteq)$, discussed in the example above.
- If $(\mathrm{E}_\mathrm{j}^1 \sqsubseteq \mathrm{E}_\mathrm{w}^1$ and $(x_{u-w} \sqsubseteq \mathrm{E}_\mathrm{w}^1$ and $(x_{i-j} \sqsubseteq \mathrm{E}_\mathrm{j}^1$ *or* $x_{i-j} \equiv \mathrm{E}_\mathrm{j}^1)))$ then we cannot assume any constraint.
- If $(\mathrm{E}_\mathrm{j}^1 \equiv \mathrm{E}_\mathrm{w}^1$ and $(x_{i-j} \equiv \mathrm{E}_\mathrm{j}^1$ and $x_{u-w} \equiv \mathrm{E}_\mathrm{w}^1))$ then $constr(x_{i-j}, x_{u-w}, \equiv)$. This constraint drives different methods that compute equivalences to reach an agreement concerning the mapping of each class.
- If $(\mathrm{E}_\mathrm{i}^1 \equiv \mathrm{E}_\mathrm{w}^1$ and $(x_{j-i} \sqsubseteq \mathrm{E}_\mathrm{i}^1$ and $x_{u-w} \equiv \mathrm{E}_\mathrm{w}^1))$ then $constr(x_{j-i}, x_{u-w}, \sqsubseteq)$.
- If $(\mathrm{E}_\mathrm{i}^1 \equiv \mathrm{E}_\mathrm{w}^1$ and $(x_{j-i} \equiv \mathrm{E}_i^1$ and $x_{u-w} \sqsubseteq \mathrm{E}_\mathrm{w}^1))$ then $constr(x_{u-w}, x_{j-i}, \sqsubseteq)$.
- If $(\mathrm{E}_\mathrm{i}^1 \equiv \mathrm{E}_\mathrm{w}^1$ and $(x_{j-i} \sqsubseteq \mathrm{E}_\mathrm{i}^1$ and $x_{u-w} \sqsubseteq \mathrm{E}_\mathrm{w}^1))$ then we cannot assume any constraint.

It must be pointed out that we deal only with constraints that are generated from subsumees to subsumers (i.e., $\mathrm{E_i} \sqsubseteq \mathrm{E_w}$), and not in the reverse order: The latter choice would generate much more constraints, and therefore, the number of agents' neighbors would increase dramatically, thus reducing the efficiency of the max-sum algorithm.

Constraints specify how the utility of each agent is affected by its own state and the state of its neighbors. The utility function of an agent A_{j-i}—corresponding to the class $\mathrm{E_i}$ of the base ontology and to the alignment method AM_j—is given by the following formula:

$$U_{j-i}(X) = n\gamma_{j-i}(x_{j-i}) - \sum_{x_{k-l} \in N(U_{j-i})} \sum_{x_{u-w} \in C(x_{k-l}, x_{j-i})} x_{k-l} \otimes x_{u-w}$$

$$x_{k-l} \otimes x_{u-w} = \begin{cases} a & if\ constr(x_{k-l}, x_{u-w}, \equiv)\ \ but\ x_{k-l} \not\equiv x_{u-w} \\ b & if\ constr(x_{k-l}, x_{u-w}, \subseteq)\ \ but\ x_{k-l} \not\subseteq x_{u-w} \\ 0 & \text{otherwise.} \end{cases}$$

In all conducted experiments, parameters a and b are set to 1 and 0.2, respectively. Moreover, $n\gamma_{j-i}(x_{j-i})$ is the normalized confidence by which the method AM_j maps the class $\mathrm{E_i}$ of the base ontology to a class of the other ontology assigned to x_{j-i}. Confidence values of mappings computed by AM_j (i.e., of values in D_{j-i}) are being normalized by dividing them by the $max_{Xj-i \in Dj-i}(\gamma_{j-i}(x_{j-i}))$. $N(U_{j-i})$ is the set of the variables connected to the utility U_{j-i}, and $constr(x_{k-l}, x_{u-w}, \mathrm{r})$ denotes the constraint which states that x_{k-l} must be related via r with x_{u-w}. In case a method does not assign a confidence value to any of the *assessed* mappings, the confidence for these mappings is set to be equal to 1. For the mappings that a method does not assess to hold, the confidence value is set to be 0.

Finally, $C(x_{k-l}, x_{j-i})$ is a set of variables that depend on two given variables, namely x_{k-l}, x_{j-i} and is defined as follows:

$$C(x_{k-l}, x_{j-i}) = \{x_{z-w} \in N(U_{j-i}) | idx(x_{z-w}) > idx(x_{k-l}) \\ and\ [x_{k-l} \in N(U_{z-w})\ or\ x_{z-w} \in N(U_{k-l})]\},$$

where *idx(x)* is an integer that is assigned to a variable by the order in which the corresponding agent is created in the factor graph.

It must be noticed that according to the above specifications, the utility function of an agent depends on the states of its immediate neighbors, as well as on the states of the neighbors' neighbors, based also on an assignment of preferences to the states of these agents: According to our experimental results, which are in compliance with results reported in [9], making the utility depend solely on the states of neighbor agents is not sufficient for the algorithm to effectively prevent cycling, causing both the messages and the preferred states of the agents not to converge.

The utility function specified above allows agents to take into account not only conflicts with their neighbors, but also conflicts among these neighbors, preventing cycling and driving the algorithm to convergence: While variables converge when they reach a fixed state, messages converge when these are the same with ones send via a specific edge in a previous cycle of execution.

In the current implementation of the algorithm, at each execution cycle agents are ordered randomly, update their messages and state based on the previous messages that they had received, and then propagate their outgoing messages to their neighbors. Therefore, this implementation of the max-sum algorithm simulates asynchronous message passing and the distribution of calculations.

5 Experimental Results

To study whether and how the proposed synthesis method satisfies our requirements set in the introduction, we have performed a series of experiments using three individual methods: The "Classification-Based Learning of Subsumption Relations for the Alignment of Ontologies" method, a Vector-Space Model-based method, and the COCLU lexical matching method. These methods have been chosen due to the different types of mapping relations they compute, due to the different types of information they exploit, as well as due to their fundamentally different approach to the alignment of ontologies. It must be pointed out that our purpose is not to construct a very effective alignment method per se, but to study the potential of the generic synthesis method proposed to satisfy the desiderata specified in the introduction of this paper.

In the paragraphs that follow, we provide background information on each of the synthesized methods. Furthermore, we define a baseline synthesis method so as to compare the experimental results of the proposed synthesis method with those of the baseline method, when both methods synthesize the three above-mentioned individual methods.

5.1 The Individual, Synthesized Methods: CSR, VSM, and COCLU

The "Classification-Based Learning of Subsumption Relations for the Alignment of Ontologies" method [11] computes subsumption relations between class pairs of

two distinct ontologies by means of a classification task, using state-of-the-art supervised machine learning methods. The employed classification mechanisms exploit two types of classes' features: Class properties or words extracted from labels, comments, properties, and instances of classes [11]. Class pairs are represented as feature vectors of length equal to the number of the distinct features of the source ontologies. The examples for the training of the classifiers are being generated by exploiting the known subsumption and equivalence relations in both source ontologies, considering each source ontology in isolation. The confidence value $\gamma_{csr\text{-}i}(x_{csr\text{-}i})$ computed by the CSR method ranges in {0, 1} depending on CSR's assessment on whether the class $\mathtt{E}_\mathtt{i}$ from the base ontology subsumes the class assigned to $x_{csr\text{-}i}$.

The Vector Space Model-based method [5], computes the matching of two pseudo documents. In our case, each such document corresponds to a class and it is produced by the words in the vicinity of each ontology class. The "vicinity" of a concept includes all words occurring 1) to the local name, label, and comments of this concept, 2) to any of its properties (exploiting the properties' local names, labels, and comments), as well as 3) to any of its related concepts or instances, as is detailed in [5].

Each document is represented by a vector of n weighted index words, where the weight of a word is the frequency of its appearance in the document. The similarity between two vectors is computed by means of the cosine similarity measure. Therefore, the confidence value $\gamma_{vsm\text{-}i}(x_{vsm\text{-}i})$ computed by the VSM method ranges in [0, 1] and it equal to the cosine similarity of the vector representation of a class $\mathtt{E}_\mathtt{i}$ in the base ontology and of any class assigned to $x_{vsm\text{-}i}$.

COCLU was originally proposed as a method for discovering typographic similarities between sequences of characters over an alphabet (ASCII or UTF character set), aiming to reveal the similarity of classes instances' lexicalizations during ontology population [1]. It is a partition-based clustering algorithm which divides data into clusters and searches the space of possible clusters using a greedy heuristic. Each cluster is represented by a model realized by a Huffman tree which is incrementally constructed as the algorithm dynamically generates and updates the clusters by processing one string (instance's surface appearance) at a time. The use of a model classifies the algorithm to the model-based learning algorithms. To decide whether a new string should be added in a cluster (and therefore, that it lexicalizes the same class as the other strings in the cluster do) the algorithm employs a score function that measures the compactness and homogeneity of a cluster. This score function, Cluster Code Difference (*CCDiff*), is defined as the difference of the summed length of the coded string tokens that are members of the cluster, and the length of the cluster when it is updated with the candidate string. This score function groups together strings that contain the same set of frequent characters according to the model of a cluster (e.g., Pentium III and PIII). According to the above, COCLU takes as input two strings and returns their similarity. The local name, label, or comment of a class (considered as a string), belongs in a particular cluster when the CCDiff is below a specific threshold (set to 0.98 in all conducted experiments) and it is the smallest between the CCDiff's of the given string and all existing clusters. A new cluster is created if the candidate string cannot be assigned to any of the existing clusters. As a result, it is possible to use the algorithm even when no initial clusters are available. The confidence value $\gamma_{coclu\text{-}i}(x_{coclu\text{-}i})$ computed by the COCLU method is equal to the CCDiff of the class $\mathtt{E}_\mathtt{i}$ in the base ontology and any class assigned to $x_{coclu\text{-}i}$, and ranges in [0, 1].

Summarizing the above, CSR computes only subsumption relations between the classes of the input ontologies, excluding equivalences between classes. The information that CSR exploits concerns either terms in the vicinity of classes' specifications or classes' properties. On the other hand, VSM exploits only the terms in the vicinity of classes, aiming to compute equivalences among classes of distinct ontologies. In contrast to both of the above approaches, COCLU exploits the similarity of classes' lexicalizations, labels, and comments. COCLU computes equivalence relations between classes.

5.2 The Baseline Synthesis Method

The proposed synthesis method is juxtaposed to a simple synthesis method that maps each class in the base ontology to the class that, given the confidence values from all individual methods, scores the highest geometric mean value, among all the classes of the nonbase ontology.

Specifically, given two input ontologies O_1 and O_2, where O_1 is the base ontology, and a class $\mathtt{E}_\mathtt{i}^1$ from O_1, let D_{j-i} be the set of candidate mappings for $\mathtt{E}_\mathtt{i}^1$ using the mapping method AM_j. If AM_j does not locate any mapping for $\mathtt{E}_\mathtt{i}^1$, then D_{j-i} is empty.

The implemented baseline synthesis method computes for each class $\mathtt{E}_\mathtt{i}^1$

$$\mathrm{x} = \mathrm{argmax}\, Geometric\_Mean(conf_1(x), conf_2(x), \ldots),$$

where $x \in \cup_j D_{j-i}$.

The function $Geometric\_Mean(value_1, \ldots, value_k)$ calculates the geometric mean of k given values, and $conf_j(x)$ is either 1) the confidence value $\gamma_{j-i}(x)$ computed by the mapping method AM_j, given that the class assigned to x is in D_{j-i}, or 2) zero, given that the value assigned to x is not in D_{j-i}.

5.3 The Data Set

The testing data set has been derived from the benchmarking series of the OAEI 2006 contest [8]. We use this data set, since based on this we have constructed gold standard mappings for the CSR method. Additionally, as the CSR method exploits class properties (in cases where properties are used as class pairs' features) and hierarchies of classes (for the generation of training examples), this data set does not include OAEI 2006 ontologies whose classes have no properties or there are no hierarchical relations between classes. The compiled corpus is available at the URL "http://www.icsd.aegean.gr/ai%2Dlab1/csr/index.htm." For each pair of ontologies, we have created the gold standard ontology, including subsumption and equivalence relations among classes.

All benchmarks (101-304) except 301-304, define the second ontology of each pair as an alteration of the base

TABLE 1
The Data Set

Ontology	Names	Comments	Hierarchy	Instances	Properties	Classes	Comment
101	-	-	-	-	-	-	Base Ontology.
103	-	-	-	-	-	-	Generalization in OWL Lite. OWL constructs owl:unionOf, owl:oneOf and the property types (e.g. owl:TransitiveProperty) have been discarded.
104	-	-	-	-	-	-	Restriction in OWL Lite. Unavailable constraints have been discarded.
201	N	-	-	-	-	-	No names.
202	N	N	-	-	-	-	No names, no comments.
203	-	N	-	-	-	-	No comments.
204	C	-	-	-	-	-	Naming changes.
205	S	-	-	-	-	-	Synonyms.
206	F	F	-	-	-	-	Translation.
207	F	-	-	-	-	-	Translation.
208	C	N	-	-	-	-	Naming changes, no comments.
209	S	N	-	-	-	-	Synonyms, no comments.
210	F	N	-	-	-	-	Translation, no comments.
222	-	-	F	-	-	-	Flattened hierarchy.
223	-	-	E	-	-	-	Expanded hierarchy.
224	-	-	-	N	-	-	No instances.
225	-	-	-	-	R	-	No local restrictions on properties.
230						F	Flattened Classes.
231	-	-	-	-	-	-	No Alteration.
237	-	-	F	N	-	-	Flattened hierarchy, no instances.
238	-	-	E	N	-	-	Expanded hierarchy, no instances.
249	N	N	-	N	-	-	No names, no comments, no instances.
251	N	N	F	-	-	-	Flattened hierarchy, no names, no comments.
252	N	N	E	-	-	-	Expanded hierarchy, no names, no comments.
258	N	N	F	N	-	-	Flattened hierarchy, no names, no comments, no instances.
259	N	N	E	N	-	-	Expanded hierarchy, no names, no comments, no instances.
301	Has names.	Has comments.	Flat hierarchy (one root concept with child concepts, i.e., depth =1).	No instances.	Has properties.	Has classes.	Real World Ontology: BibTeX/MIT.
302	Has names.	No comments.	Has hierarchy (depth=2).	No instances.	Has properties.	Has classes.	Real: BibTeX/UMBC.
303	Has names.	No comments.	Has hierarchy (depth=4).	No instances.	Properties are defined only in root concepts	Has classes.	Real: Karlsruhe.
304	Has names.	Has comments.	Has hierarchy (depth=3).	No instances.	Some concepts have no related properties.	Has classes.	Real: INRIA.

*Name: Name of entities is replaced by random strings (N), synonyms (S), different name changes (C) (*e.g. uppercasing, underscore*), or strings in another language than English (F).*
Comments: Comments are suppressed (N) or translated in another language than English (F).
Hierarchy: is expanded (E) in depth or flattened (F).
Instances: are suppressed (N).
Properties: are suppressed (N) or all local restrictions on properties are suppressed (R).
Classes: are expanded (E), i.e. one class is replaced by several classes.

ontology numbered 101. Table 1 summarizes the alterations performed on each ontology.

5.4 Experimental Results and Discussion

In this section, we present the results of the conducted experiments. Specifically, we present the results for each individual method when it performs individually and when it performs in synthesis with other methods, so as to show how the synthesis affects methods' individual performance. Specifically, we synthesize the VSM, COCLU, and CSR methods presented in Section 5.1. Moreover, we present the results achieved by the baseline method so as to compare them with those achieved by the synthesized individual methods. It must be pointed out that we do not present the precision and recall of the synthesis method, as our aim here is not to propose a specific alignment method per se, but a synthesis method that specifies specific requirements. However, the set of mappings produced by the proposed synthesis method is the union of all the mappings produced by the individual methods, given that the max-sum algorithm has converged, and all stated constraints have been satisfied (i.e., all conflicts have been resolved).

Before proceeding to the experimental results concerning the effectiveness of the synthesis method, we need to comment on the convergence of the max-sum algorithm. As already pointed out, the utility function chosen prevents cycling and drives the algorithm to convergence. This is evident by the graphs in Fig. 3 where the average number of conflicts for all experiments contacted is shown to be reduced over execution cycles.

Specifically, Fig. 3a shows the average number of conflicts concerning 1) equivalence constraints, 2) subsumption constraints, and the average number of conflicts for all constraints. Fig. 3b shows the corresponding variances: The large variance of the conflicts concerning subsumption constraints is due to cases where hierarchies are flattened (specifically, to ontologies 222, 237, and to ontology 301) and to ontology 303 where properties are defined only in root concepts: In these cases, CSR fails to perform adequately [11].

As Fig. 3 shows, the algorithm reaches quality solutions in terms of the number of conflicts of the final solution, in a small number of execution cycles: Having noticed that the performance of the algorithm has no significant improvement after the ninth execution cycle (even if the conflicts have the potential to be reduced), in the experiments contacted, we have set the max-sum algorithm to stop after the 10th execution cycle. During this time, agents exchange 298 messages in total.

Concerning the number of conflicts over execution cycles, the max-sum algorithm outperforms other approximation algorithms that exploit information about the

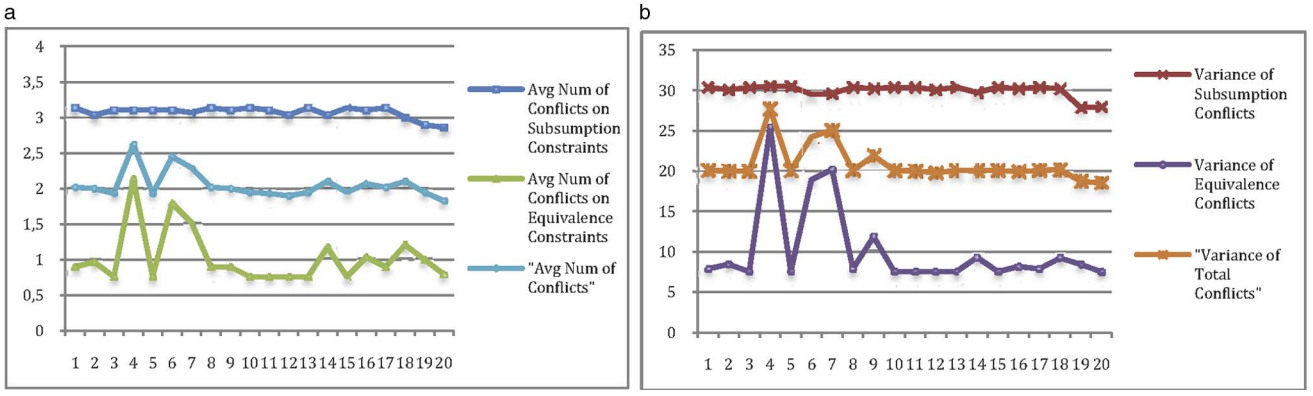


Fig. 3. (a) Average number of conflicts per execution cycle for all the experiments and (b) Variance of the number of conflicts per execution cycle.

preferred states of their neighbors. This is due to the fact that the max-sum algorithm exploits information concerning the utility of neighbors [9].

Concerning the effectiveness of the synthesis method, Figs. 4 and 5 present the precision and recall of VSM and COCLU when they are applied to the benchmark series individually, and the precision and recall of the synthesized VSM, COCLU, and CSR using the max-sum algorithm. Also the precision and recall achieved by the baseline synthesis method is presented. Precision is the ratio *#correct_pairs_computed/#pairs_computed* and recall is the ratio *#correct_pairs_computed/#pairs_in_gold_standard*.

A first observation is that the effectiveness (precision and recall) of VSM is positively influenced when it is synthesized with COCLU and CSR. Specifically, on average, the precision is improved from 85 percent with a variance of 0.02 (VSM) to 93 percent with variance 0.01 (VSM Synthesized) and the recall from 66 percent with variance 0.1 to 70 percent with the same variance. Indeed, there is not any pair of ontologies in Fig. 4 and 5 where the synthesized VSM method is worst in performance than the individual VSM method. On the other hand, we observe that the effectiveness of COCLU is not improved, but only slightly in the case 223. This is due to the following reasons:

1. COCLU operates with a rather strict threshold (equal to 0.98). This implies that for each class of the base ontology, the method computes one or two candidate mappings with a similarity value above the threshold. These mappings have similarity values that are too close to each other (e.g., 0.986 and 0.99). However, given the high precision of the method, the classes with the largest similarity are indeed equivalent in the vast majority of the experiments depicted in Figs. 4 and 5 (average precision 99 percent with variance 0.0006). Due to these facts, it is very difficult for COCLU to improve and quite easy to be affected by the other methods toward erroneous decisions (e.g., in cases 301 and 304). The latter happens only in a small number of experimental cases, but it slightly affects the performance of COCLU when it performs in synthesis with the other methods.
2. As already explained in the Section 4.1, for any of the synthesized methods it holds that in case there are no candidate mappings for a specific class of the base ontology, then no agent is created for this class. In this case, the proposed synthesis method corrects only false mappings. Given the high precision of COCLU, the false mappings that may be corrected are really very few.

On the other hand, VSM, when applied individually, computes a larger number of candidate mappings and more false mappings than COCLU (its average precision is 85 percent). This leaves much space for improvement in the performance of VSM when it is synthesized with other methods.

Of high interest are the cases where the altered ontology is expanded (e.g., 223). There, we observe that the synthesized VSM and the synthesized COCLU methods perform better than the corresponding methods when applied individually. This is true for both measures: precision and recall. This is due to the large number of constraints that exist among the agents, which is due to the number of subsumption relations among classes. These

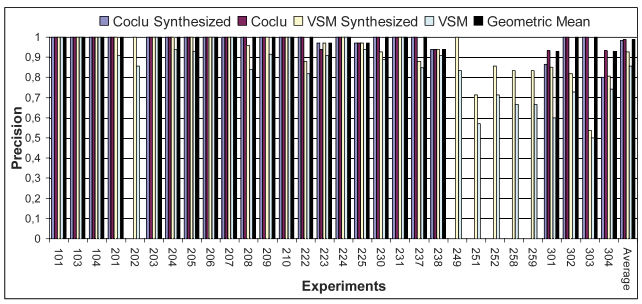


Fig. 4. Synthesis of VSM, COCLU, and CSR Methods—Precision.

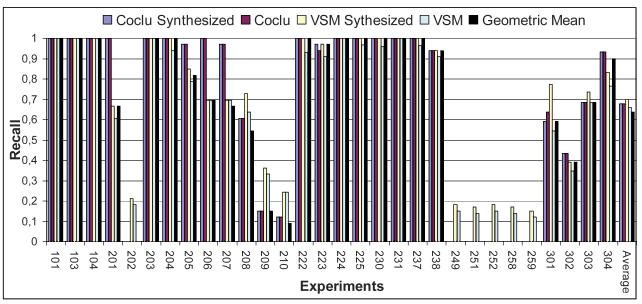
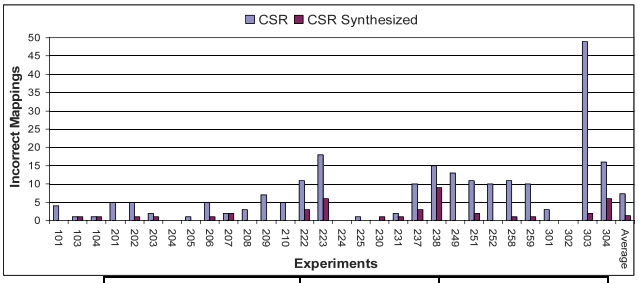


Fig. 5. Synthesis of VSM, COCLU, and CSR Methods—Recall.



	CSR	CSR Synthesized
Incorrect mappings [Average (Variance)]	7.6 (90)	1.6 (6)

Fig. 6. Synthesis of VSM, COCLU, and CSR Methods—Incorrect mappings by CSR.

constraints guide the synthesized methods to select among the candidate mappings those that are consistent among themselves, maximizing also the social welfare. In addition, we observe that in the experiment 223 the precision of COCLU is 94 percent, and thus there is space for improvement by correcting false positives: Indeed, it improves by 3 percent. This also provides evidence toward the conjecture that in order to further improve the proposed synthesis method and further guide individual methods to improve themselves, more types of constraints should be introduced. This may be done by introducing constraints among other ontology elements, beyond classes (e.g., by exploiting the domain and range of classes' properties).

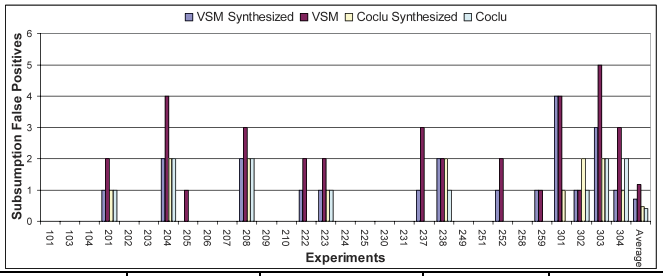
Concerning the effect of the synthesis on the CSR method, we present the number of assessed mappings that CSR incorrectly assesses to be correct. It is important to state that we cannot compare the precision and recall of CSR when applied individually to its synthesized version: This is so, since when CSR is applied individually it locates one-to-many (1:n) subsumption mappings, while when it is being synthesized using the proposed method it is "forced" to compute one-to-one (1:1) subsumption mappings. Furthermore, concerning precision, CSR locates mappings with confidence value either 1 or 0. As a result, the synthesized CSR method selects a mapping among those computed (with confidence set to 1), based only on the minimization of the conflicts introduced in the factor graph. For example, if for a class of the base ontology CSR finds five mappings and only one introduces a conflict, then the synthesized CSR will not be able to distinguish which one of the other four to select (given that the same set of constraints hold for these values). For these reasons, in Fig. 6, we present the number of wrongly assessed mappings. The table following the graph shows the average score for all experiments, together with the variance. Results show that the incorrect mappings produced by the synthesized CSR in comparison to those computed when CSR performs individually are significantly reduced in all cases, with a significantly reduced variance as well.

Fig. 7 depicts the number of class pairs that VSM and COCLU incorrectly assess to be related via the equivalence relation instead of the (correct) subsumption relation. These results show that the synthesis reduces the number of the incorrect mappings for VSM, but due to the reasons explained above, it slightly reduces the performance of COCLU when this method performs in synthesis with VSM and CSR. This is important, since state-of-the-art systems tend to confuse subsumption relations with equivalence ones [12]. As depicted in Fig. 7, incorrect mappings are reduced for the synthesized methods in average, with a reduced variance. This is the case for all experimental cases where incorrect mappings do never increase.

The experiments which provide evidence for the high tolerance of the proposed method to failures of the individual methods are 202, 249, 251, 252, 258, and 259. In these cases, only instances are available (classes have no labels or comments) and in some of these cases the ontologies are expanded or flattened. We observe that although COCLU completely fails (0 percent in both precision and recall), the synthesized VSM is substantially positively influenced by CSR in both terms of precision and recall. Specifically, precision has an increase of 14-17 percent and recall increases from 3 percent up to 5 percent. We point out that the baseline synthesis method scores 0 percent in both precision and recall, a fact that shows its sensitivity in any of the synthesized methods: If one of the methods fails, then the baseline synthesis method fails as well. The same phenomenon occurs even if we apply a smoothing function to eliminate zeros: The baseline method achieves precision and recall that are very close to zero.

6 Concluding Remarks and Future Work

In this paper, we tackle the problem of effectively synthesizing different ontology alignment methods. For this purpose, we propose a "model-based synthesis"



	Coclu	Coclu Synthesized	VSM	VSM Synthesized
Subs False Pos [Average (Variance)]	0.4 (0.5)	0.46 (0.6)	1 (2)	0.7 (1)

Fig. 7. Synthesis of VSM, COCLU, and CSR Methods—Incorrect Subsumption Mappings.

method, which addresses this problem by maximizing the social welfare within a group of interacting agents (i.e., maximizing the sum of utilities of individual agents). Each agent is responsible for making an alignment decision concerning a specific ontology element using a specific alignment method. Agents need to reach an agreement to the mapping of the ontology elements, consistently to the semantics of specifications and according to their preferences, which are being provided by the alignment methods.

To sum up, the proposed synthesis approach addresses all the requirements for a synthesis method

- *Generality.* The proposed method addresses the problem as a generic coordination problem among a number of agents. The proposed method can be used to synthesize any set of mapping methods. In case, a method does not provide a similarity value, the algorithm focuses to the minimization of violations of semantic constrains encoded in the factor graph.
- *Optimality.* The proposed solution aims to maximize the social welfare of agents, i.e., the sum of their utilities, taking into account the preferences of individual agents and the semantics of specifications. This modeling of the synthesis problem results to a cyclic factor graph, where as stated in the literature [9], [10] has a near-optimal solution and a very good balance between optimality and scalability.
- *Scalability.* The proposed method aims to enforce a decentralized solution, where each agent has knowledge of and can directly communicate with a small number of neighbors, which is a small proportion of the total number of agents. This is true, although, as already said, the current implementation does not distribute agents in different machines. Therefore, we are dealing with a centralized implementation of the max-sum algorithm, simulating the asynchronous, distributed calculations of agents.

The "hardest" calculation an agent does is the calculation of a "function to variable" message (3), with a complexity that is exponential to the number of neighbors the factor has. As the complexity heavily depends on the number of constraints related to an agent's function (and thus, on the number of the neighbors of each factor), we need to measure the number of constraints between variables. This number ranges in $[K, K + \lambda \times K]$, when the synthesized methods compute equivalence relations. K is the number of synthesized methods and λ is the number of classes that directly subsume the class that the agent tries to map. Usually, which is also the case in our experiments, λ is equal to 1. Methods that compute subsumptions do not affect the number of constrains (in some cases they drop the minimum number of constraints further down), as they introduce constraints concerning their own functions. For example, in Section 5, where we synthesize three methods, one that locates subsumption relations among classes (CSR) and two that locate equivalences (COCLU and VSM), the number of constraints ranges in [2], [4]

- *Expandability.* New individual methods can easily be added to the synthesis model, independently to the other synthesized methods.
- *Tolerance to failures of individual methods.* It is clear from the results commented in Section 5.4 that when one of the synthesized methods (in our case COCLU) fails to compute mappings, not only the synthesis performs as effectively as the best individual method, but it also improves the effectiveness of the individual methods.
- *Provision of feedback to the individual methods.* As commented in Section 5.4, the proposed synthesis method exploits the mappings computed by *all* individual methods to drive individual methods to improve their effectiveness by choosing mappings that are mostly preferred by them, in conjunction to being consistent with the mappings computed by the other methods: This is clearly shown in the experiments contacted.
- *Being a semantics-based approach.* As it has already mentioned, the proposed mappings must form a consistent set of mappings with respect to the semantics of ontology elements' specifications. To do so, the proposed model-based synthesis method drives mapping methods to compute mappings that respect a set of validity constraints.

To the best of our knowledge, there is no method that performs such a synthesis, but there are some alignment methods with similarities to the one we propose.

The OMEN system [13] models pairs of concepts (i.e., possible mappings) as nodes of a Bayesian network, where the edges among these nodes capture the dependences between them. Although Bayesian networks (BN) are related to factor graph (actually a BN is a special case of a FG), OMEN focuses on the propagation of similarity (location of new mapping pairs) and not on the "model-based synthesis" of the mapping pairs located by the synthesized methods.

Following the same intuition with the OMEN system, the GLUE system [14] employs the Relaxation Labeling technique for achieving the same purpose. Another interesting approach is presented in [15], where the authors propose a Bayesian network-based modeling of the interdependencies among different ontology mapping methods. This approach does not focus to the synthesis of the mapping methods, but to the discovery of their interdependencies (e.g., if a specific method is redundant).

Aiming to bridge the semantic gap among agents conceptualizations, Laera et al. [16] propose the exploitation of given ontology mappings (mappings between ontology classes) via agents' negotiation techniques. Our proposed approach focuses to proposing a synthesis approach where neighboring agents interact in order to produce an agreed set of mappings between ontology classes. Similarly to our approach, the authors of the work [17] propose an argumentation-based negotiation framework of interacting agents for filtering given mappings. In more detail, an agent is created for each given candidate mapping. Agents exchange arguments based on the semantics of the input ontologies and various ontology elements' similarity measures (e.g., similar labels, similar structure), while trying to reach a consensus. The nonrejected candidate mappings are returned as output of the method.

Future work concerns the use of this approach in an inherently distributed setting where each agent has its own ontology and needs to communicate consistently with others: In such a case, agents need to agree on the set of mappings between their ontology elements in a consistent way. This is a problem that is different from the one discussed in this paper, although it is closely related to it: Each agent may have the capability to produce mappings using a specific alignment method, but it may not completely know the ontologies of its neighbors, or of the other distant agents in a cycle. Indeed, missing information as well as propagation of results to distant agents are important issues that need to be taken into account for computing consistent solutions involving all related agents. Furthermore, in the synthesis model presented here, agents correspond to the elements of the "base" ontology: In a distributed setting, variables of different agents correspond to elements of different ontologies. Therefore, interagent constraints may need to be more complex than those discussed in this paper.

Beyond dealing with this problem, future work on synthesizing different ontology alignment methods will mainly be focused on the incorporation of more types of constraints.

ACKNOWLEDGMENTS

This research project has been cofinanced by E.U.—European Social Fund (75 percent) and the Greek Ministry of Development-GSRT (25 percent). The authors also want to thank Dr. A. Farinelli for his insightful hints during the implementation of the Max-Sum algorithm. This work was completed while George A. Vouros was a professor in the Department of Information and Communication Systems, University of the Aegean, Greece.

REFERENCES

[1] K. Kotis, A. Valarakos, and G.A. Vouros, "AUTOMS: Automating Ontology Mapping through Synthesis of Methods," *Proc. Ontology Matching Int'l Workshop Ontology Alignment Evaluation Initiative (OAEI),* 2006.

[2] S. Castano, A. Ferrara, and S. Montanelli, "H-Match: An Algorithm for Dynamically Matching Ontologies in Peer-Based Systems," *Proc. First VLDB Int'l Workshop Semantic Web and Databases (SWDB '03),* 2003.

[3] D. Aumüller, H. Do, S. Maßmann, and E. Rahm, "Schema and Ontology Matching with COMA++," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2005.

[4] B.T. Le, R. Dieng-Kuntz, and F. Gandon, "On Ontology Matching Problems—For Building a Corporate Semantic Web in a Multi-Communities Organization," *Proc. Sixth Int'l Conf. Enterprise Information Systems (ICEIS),* 2004.

[5] V. Spiliopoulos, A.G. Valarakos, G.A. Vouros, and V. Karkaletsis, "SEMA: Results for the Ontology Alignment Contest OAEI 2007," *Proc. Ontology Matching Int'l Workshop Ontology Alignment Evaluation Initiative (OAEI),* 2007.

[6] J. Madhavan, P. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," *Proc. 27th Int'l Conf. Very Large Data Bases,* 2001.

[7] J. Euzenat and P. Valtchev, "Similarity-Based Ontology Alignment in OWL-Lite," *Proc. 15th European Conf. Artificial Intelligence (ECAI),* 2004.

[8] Ontology Alignment Evaluation Initiative, http://oaei.ontologymatching.org/ (last accessed 31/05/2009), 2012.

[9] A. Farinelli, A. Rogers, A. Petcu, and N.R. Jennings, "Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm," *Proc. Seventh Int'l Conf. Autonomous Agents and Multiagent Systems (AAMAS '08),* 2008.

[10] F.R. Kschischang, B.J. Frey, and H.A Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Information Theory,* vol. 47, no. 2, pp. 498-519, Feb. 2001.

[11] V. Spiliopoulos, G.A. Vouros, and V. Karkaletsis, "On the Discovery of Subsumption Relations for the Alignment of Ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 8, no. 1, pp. 69-88, Mar. 2010.

[12] O. Svab, V. Svatek, and H. Stuckenschmidt, "A Study in Empirical and "Casuistic" Analysis of Ontology Mapping Results," *Proc. European Semantic Web Conf. (ESWC),* pp. 655-669, 2007.

[13] M. Prasenjit, F.N. Noy, and A.R. Jaiswal, "OMEN: A Probabilistic Ontology Mapping Tool," *Proc. Fifth Int'l Semantic Web Conf. (ISWC),* 2005.

[14] A. Doan, P. Domingos, and A. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach," *Proc. ACM SIGMOD Conf. Management of Data,* 2001.

[15] O. Šváb and V. Svátek, "Combining Ontology Mapping Methods Using Bayesian Networks," *Proc. Int'l Workshop Ontology Matching,* 2006.

[16] L. Laera, V. Tamma, J. Euzenat, T. Bench-Cappon, and T. Payne, "Reaching Agreement over Ontology Alignments," *Proc. Fifth Int'l Semantic Web Conf. (ISWC),* 2006.

[17] M. Morge and J.C. Routier, "Debating over Heterogeneous Descriptions," *J. Applied Ontology,* vol. 2, nos. 3/4, pp. 333-349, 2007.

Vassilis Spiliopoulos received the PhD degree from the Department of Information and Communication Systems Engineering, University of the Aegean, in 2009. Currently, he is working as a software engineer at Siemens Enterprise Communications. His research interests include machine learning, constraint satisfaction, ontology alignment, semantic search, and recommendation systems. Further details concerning his work can be found at http://iit.demokritos.gr/~vspiliop.

George A. Vouros received the BSc degree in mathematics, and the PhD degree in artificial intelligence both from the University of Athens, Greece. Currently, he is a professor at the University of Piraeus, Greece. He has done research in the areas of expert systems, knowledge management, collaborative systems, ontologies, and agents and multiagent systems. Further details concerning his work can be found at http://ai-group.ds.unipi.gr/georgev.