Coincidence-Based Refinement of Ontology Matching

S. H. Haeri
School of Computer Science
Department of Mathematical Sciences
Sharif University of Technology,
Tehran, Iran
shhaeri@math.sharif.edu

B. Bagheri Hariri
Semantic Web Research Laboratory
Computer Engineering Department
Sharif University of Technology,
Tehran, Iran
hariri@ce.sharif.edu

H. Abolhassani
Semantic Web Research Laboratory
Computer Engineering Department
Sharif University of Technology,
Tehran, Iran
abolhassani@sharif.edu

***Abstract*— Since the beginning of the evolution of Semantic Web in the late 90's, many different aspects of it have received a great interest in both academy and technology. Alongside, because many problems get reduced to the problem of Ontology Matching, so many researches across the world have been devoted to this. Amongst these researches, one can find a variety of techniques employed for dealing with the problem. In all such techniques, there usually exist two different phases in coming to a to-be-proposed matching; Firstly, they come to an initial estimate for the similarity of the concepts involved. Secondly, based on certain criteria, they offer a (set of) matching(s). In this paper, we are about to present a new criteria for that final phase.**

I. Introduction

Semantic Web, as like as the web itself, is by design distributed and heterogeneous. Alongside, ontology is used to support interpretability and common understanding between the different peers of the task. However, in many real cases, the ontologies themselves also suffer from some heterogeneity. It is along tackling this heterogeneity where Ontology Alignment is used for finding semantic interrelationships amongst the entities of ontologies. [1] defines ontology alignment as:

> ... given two ontologies which describe each a set of discrete entities (which can be classes, properties, rules, predicates, etc.), find the relationships (e.g., equivalence or subsumption) holding between these entities.

Many of the existing methods for ontology alignment compare similarity of entities using some predefined measures (phase 1), and via the interpretation of the results, they put forward some possible set of semantic relationships amongst the entities. Considering O_1 and O_2 as the ontologies we are about to put into alignment, define $O = O_1 \cup O_2$. With this nomination in mind, typically a similarity measure is formally defined as below [1]:

A Similarity $\delta : O \times O \to R$ is a mapping from a pair of entity to a real number - expressing the similarity between two objects such that:

$$\begin{array}{ll} \forall x, y \in O, \delta(x, y) \geq 0 & (positiveness) \\ \forall x \in O, \forall y, z \in O, \delta(x, x) \geq \delta(y, z) & (maximality) \\ \forall x, y \in O, \delta(x, y) = \delta(y, x) & (symmetry) \end{array}$$

In many methods, it is quite common to first define a set of similarity measures, and then apply them like a *compound similarity measure*. (See Fig. 1 for example.) With application of this set of (compound) similarity then, they come to an initial guess. There is afterwards another phase in which they make the final decision. In this phase, they decide on the ultimate set of satisfactory correspondence between the ontologies. From this point of view, alignment refinement (or matching refinement) is the methods for improving the quality or ease of alignment extraction (or mapping extraction). (One possible way of using refinement is like Fig. 1.)

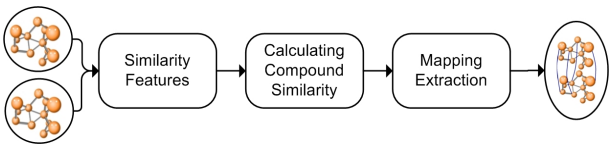

Fig. 1. A Simplified Alignment Framework

In this paper, it is first tried to justify our **new** method and the theoretical background behind it. Then, there will be a workaround for dealing with the complexity of solution. We finalise by presenting a pseudo-code for the our entire work, and apply that to an example. It is worth mentioning that our entire work presented here is original, and our method combines different ideas from different realms of science and engineering, including Ontology Matching, Graph Isomorphism, Metric Spaces, and Domain Theory.

II. Related Works

As it is listed in [1], the works on the alignment extraction phase are not so many. Ehrig and Sure [2] present a variety of threshold-based tricks for that, and Valtchev [3] considers this problem as an optimization one, and offers a solution based on this interpretation. In [4], on the other hand, the *stable marriage problem* ([5] and [11]) is exploited for coming to a more reasonable extraction. However, in all of the above works, the sole of the ontologies as **taxonomies** is neglected.

Let's put it this way: Ontologies, from the graph theoretical viewpoint, are directed acyclic graphs [12] with edges having types. To the knowledge of the authors, no work is so far

done on the problem of Ontology Alignment or Ontology Matching in which this graph theoretic backbone of problem is scrutinised. In fact, all the existing works have neglected the fact that a **single** ontology is inherently a **structure**, and has **interconnections** between its own concepts.

We believe that there is a vast area for new works in Semantic Web for adding to the precision of the existing works, based on restoring this backbone. This paper is indeed trying to leverage that for a special area in Semantic Web, which is Ontology Matching. Unlike all of the above works, we give this backbone a great role in the extraction phase. This role is in fact our new method of matching refinement.

III. Justification of our Method and its Background

In this section, we first give an intuition for our method, take a look to the theoretical background. Finally, a precise specification of the problem using this background is presented.

A. Intuition

Before saying exactly how it is that we care about the structure of ontologies, whilst the rest do not: For one moment, forget about ontology matching, and consider the following basic-geometry problem: When do you call a pair of triangles **the same**? When they are equal in the geometric sense? You mean you consider the two triangles in Fig. 2.a **the same**? We do doubt [1]! Now, take a look to Fig. 2.b. (The solid lines indicate one triangle, the dotted ones indicate another, whilst the vertices of the triangles coincide.) Up to our understanding, we - human-beings - consider these two triangles **the same**! Now consider the case of Fig. 2.c and Fig. 2.d in comparison; trying to give a fuzzy interpretation to the concept of "being the same" - or **coincidence**, it should be said that: the two triangles in Fig. 2.d are more the same than that of Fig. 2.c! And, the two of Fig. 2.b coincide even more.

This is what we are about to inject in the world of ontology extraction. That is, given that the phase one of ontology alignment gives us a measure for similarity across the ontologies, we consider this measure as an estimate for the distance between each pair of point, and suite it for estimating the extent to which the two ontologies - as the whole graphs - coincide. Alongside, we first offer an estimate for the extent of coincidence between two edges, and then accumulate all these as our final estimate for the coincidence of the two ontologies.

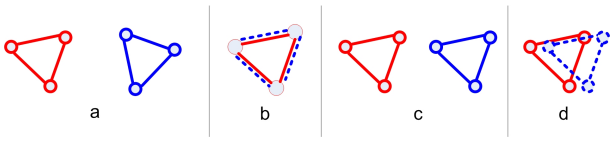

Fig. 2. Matching of Shapes

[1]In mathematical topology, these two triangles are the same in that there is a continuous bijection between them, inverse of which is also continuous.

B. Theoretical Background

As also mentioned in the previous sections, it is assumed here that, in phase one, based on certain methods, we have come to a matrix (d) containing initial guesses for the similarity of the concepts involved. Looking more precisely would reveal the fact that this gives us a *Metric Space* [2], d is the *metric* of which. [6] defines a metric space as:

> A set X, whose elements we shall call points, is said to be a metric space if with any two points p and q in X there is associated a real number $d(p, q)$, called the distance from p to q, such that:
>
> - a. $d(p, q) > 0$ if $p \neq q; d(p, p) = 0$; [self-distance]
> - b. $d(p, q) = d(q, p)$; [symmetry]
> - c. $d(p, q) \leq d(p, r) + d(r, q), for\ any\ r \in X$. [triangular inequality] [3]
>
> Any function with these three properties is called a distance function, or metric.

Another piece of theory which is of help is the notion of *Typed Graphs* [4]. In general, we call a graph G *typed* if each edge of it has a type. In other words, let's formally define $G(V, E, T)$ a typed graph if E is of type $V \times V \rightarrow T$, in which T is a set of predefined types. An edge e of type t is written $e : t$. An isomorphism from a typed graph $G(V, E, T)$ to a typed graph $G'(V', E', T)$ is a one-to-one correspondence between V and V'. We will call an edge $e(a, b) : t \in E$ *preserved under* m iff there is an edge $e'(m(a), m(b)) : t \in E'$. If both a and b get mapped to some vertex in V', yet there is no edge of type t between $m(a)$ and $m(b)$, we write $TP(e, m)$. We will call a typed graph $G(V, E, T)$ vertices of which are points in (X, d) *embedded on* X, and write $G(V, E, T, X, d)$.

And, the final piece of theory is the notion of *partial order*. [13] defines a *Partially Ordered Set* like this:

> A set P with a binary relation $\sqsubseteq$ is called *a partially ordered set* or *poset* if the following holds for all $x, y, z \in P$:
>
> 1) $x \sqsubseteq x$ (Reflexivity)
> 2) $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$ (Transitivity)
> 3) $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x \sqsubseteq x$ (Antisymmetry)
>
> We add that $\sqsubseteq$ above is called a partial order.

C. Theoretical Specification of Problem

First of all, let's mention once more that – although some experts may consider our work a method for extraction – we believe that this work offers a new criteria which helps **deciding on extraction**, as opposed to extraction itself. Secondly, let's formulate the problem of Matching Refinement using our theoretical background:

Input: A pair of ontologies, and a matrix, rows and columns of which stand for concepts from one ontology, and concepts from the other, respectively. Each cell shows the distance between the corresponding concepts.

[2]Not exactly a metric space! Please see the commentary section.
[3]Please see the commentary section for the notes on this property.
[4]There is no consensus in mathematics on this name.

From our point of view, this input is interpreted as a pair of directed acyclic graphs embedded on a metric space. So, naming the input ontologies O and O', we do not distinguish them from $G(V, E, T, X, d)$ and $G'(V', E', T, X, d)$ respectively.

Output: A ranking of possible matchings which can be a help for better extraction. From our point of view this is a partial order on the possible isomorphisms between G and G'. That is a mapping between V and V'.

To produce the above output, this paper first enumerates a list of rationales for the above partial order, and then presents one possible candidate for that. We will then discuss on possible axes along which one can tune our proposed ordering.

D. Properties of the Desired Partial Order

Here will be a set of properties which we believe any partial order for matching refinement should convince, along with our reasons. Our proposed partial order is in fact a *weight function* for matchings, so hereafter we use weight in place of it. The set of properties are divided into 6 categories, based upon preservation of the edge (under the mapping), and upon the distance between its heads.

Please note that in all of the following figures, O and O' are the inputted ontologies, a and b will be concepts in O, and, a' and b' concepts in O'. The closer a pair of concepts is depicted in figures, the closer the concepts are intended to be in the (X, d). (That is, the closer a and a' are shown in the figures, the smaller $d(a, a')$ is.) We do not force the ontologies to be disjoint, so, in each figure, you can see that the surface of ontologies may overlap. Furthermore, in each figure, the arrows show the mapping. (That is, the source of arrow is intended to be said is mapped to its destination.) And, the lines – be it solid or dotted – show the edges of graphs. (Solid lines show the edges between a and b, and dotted edges show the edges between a' and b'.)

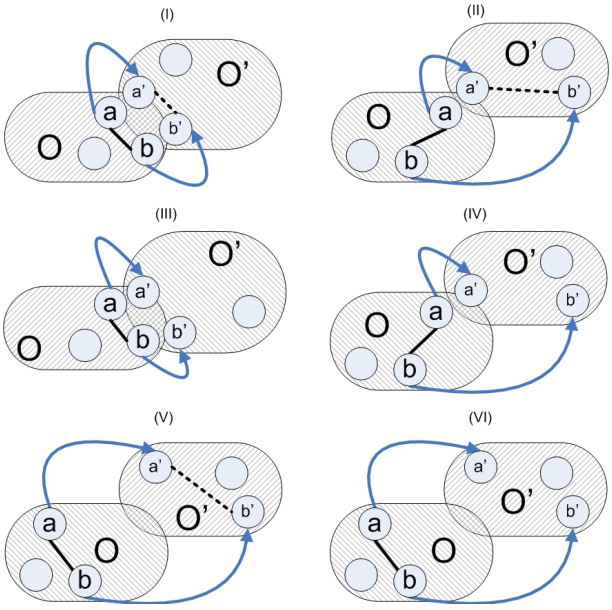

Fig. 3. Properties of Metrics

Category I. Here, a and a' are too close, like b and b'. The fact that (a, b) is preserved is of much importance to us because it means that the two **edges** coincide too much. So, we want this preserved edge to bring a great weight. If you are not justified on this, consider the case when a and b are "Animal" and "Jaguar" respectively, and a' and b' are "Living Creature" and "Tiger". The fact that there is an edge (of type *redfs:type*) between both a and b, a' and b', means very much that the two Ontologies are perhaps describing the same world.

Category II. In this category, the edge is preserved, but only one peer of the edge is close to its image. As an example of such cases, one can consider O be describing a Zoo, and O' a Museum. Furthermore, suppose that a and b are "Elephant" and "4-legged", and, a' and b' are "Mammoth" and "Ancient Creature". An interpretation of this is that although O and O' are describing two different worlds, they are perhaps getting close "from the side of a". Therefore, we would like in such cases to get a large weight, yet smaller than the previous case.

Category III. The third category is the one where an edge is not preserved whilst the relevant concepts are so close. Consider, e.g., when O is describing the Glazing Technology, whilst O' is the ontology of a simple glasses manufacturing studio. In retrospect, a and b could be "Glass" and "Frame", and a' and b' the same respectively. Of course $d(a, a')$ and $d(b, b')$ may both be very small here. We consider the non-preservation of edge a negative point, but because the vertices coincide, let us not to penalise this matching that much. This is logical because the closeness of (a, a') and (b, b') means that the edge (a', b') is perhaps mistakenly missed.

Category IV. Next, we come to the category where an edge is not preserved, whilst only one side of the edge is too close to what it is mapped to. A mapping which has done this is perhaps trying to make a mistake, but not as big as category VI. So, we will not penalise it that much. As an example of such a case, consider this: O is describing a glasses manufacturing studio, and O' is a car factory. Assume that a is "Glasses" and a' is "Glass", b could be "Frame", whilst b' is "Chassis". Like category III which is somehow dual of category I, this category can be considered dual of category II.

Category V. and VI. If the edge is preserved, although this increases the likelihood of preservation of the whole shape, if the pairs of concepts are not close at all, this should not look like a great success because the two edges are not that much coincident. In other words, although the preservation of shape is important, we do not care it that much if the edges coincide at neither end. For an example of when this looks rational, consider the case when a is "Vehicle", b is "4-wheeled", a' is "Animal", and b' is "4-legged". Therefore, for the category V, we would like the mapping to receive a low benefit. The situation is completely similar to that of category VI, so, we do not try to justify why a mappings of that category will be penalised in a large extent.

IV. The Method

In this section, we present our proposed partial order, consider the extent to which this can be appropriate, and to tune that, offer a set of heuristics. Finally, putting all the parts together, we present the entire method.

A. Our Proposed Partial Order

Adding the fact that the weighting system is expected to be symmetric in its arguments, we observed that one possible such weighting is the following [5]: (By being symmetric in its argument, we mean $w(m(G,G')) = w(m^{-1}(G',G))$.)

$$w(m) = w_0(m) - w_l(m) - w_r(m),\ where$$

$$w_0(m) = \sum_{\substack{(v_1,v_2):t\in E \\ (m(v_1),m(v_2)):t\in E'}} \overline{f}(v_1) + \overline{f}(v_2)$$

$$w_l(m) = \sum_{\substack{(v_1,v_2):t\in E \\ TP((v1,v2),m) \\ (m(v1),m(v2)):t\notin E'}} \overline{g}(v_1) + \overline{g}(v_2)$$

$$w_r(m) = \sum_{\substack{(v_1,v_2):t\notin E \\ (m(v1),m(v2)):t\in E' \\ TP((a',b'),m^{-1})}} \overline{g}(v_1) + \overline{g}(v_2)$$

$$\overline{f}(x) = \frac{1}{f(d(x,m(x)))}$$

$$\overline{g}(x) = g(d(x,m(x)))$$

The functions f and g can be considered as **normalisation** functions. Their common property is that being restrictively increasing. Otherwise, one can always find one of the six categories above in which w will misbehave. Furthermore, f should have another property as well; its range should be outside a certain neighbourhood around origin. For the case when this will result a misbehaviour, consider a pair of ontologies across which there exists a pair of concepts with distance 0. If $f(0)$ is 0, then w will become $+\infty$, regardless of the rest of mapping. And, this obviously is a big anomaly because it will cause a big class of mappings to look the same whilst they are not inherently the same. That is, in such a case, w does not do much for a big class of mappings.

In presence of a vertex which does not get mapped to anything, all the edges from that vertex – or to it – are not preserved. In these cases, the mapping should get more weight than a mapping which has mapped such edges to edges with wrong types. To tune our formula to reflect this, virtually consider it being mapped to an imaginary vertex, existence of which does not give us any information. In this case, its distance ought to be 0 from any other concept. One can easily verify it that the above weight satisfies all the conditions enumerated. As a further benefit of our proposed weighting method, we would like to notify the following: Consider the mapping m in which a is mapped to a', b to b', with an edge between a and b, type of which is different from that between a' and b'. In this case, our weighting method would penalise m twice; once because the edge connecting a and b is not preserved, and another time for when the edge between a' and b' is not.

The special case where this will become more interesting is when $(a,b) : subClassOf$, and $(b',a') : subClassOf$. Here, our weight will recognise the fact that a mapping which maps a and b to two concepts between which there is no edge at all, is better than when they get mapped to a pair of edges where there is an edge between them **with an inverse type**.

B. Commentary

As told in footnote-5, there are cases in which what the inputted matrix gives us may not be a metric space. In fact, as told in the section for mathematical background, a metric space is needed to have symmetry. However, as listed in [7], there are schema-based matching techniques which use linguistics resources. These techniques may not convince this property. That is, for example: In the Webster Collegiate Dictionary [8], "quick" is in the 12th place in the list of synonyms of "swift", whilst "swift" is second in the list of synonyms of "quick". In such a case, the symmetry property may not hold. Therefore, what we get may be a quasi-metric space [9] rather than a metric space. However, as [1] also mentions, only few authors may consider similarity metrics which do not have symmetry. So, the existing weighting formula and the assumption with it will almost always be convincing. Even in case one is faced with an application where there inherently exists no symmetry, a little tweak in the formula will give rise to a **symmetric weighting formula** which still convinces all the conditions listed in section 4:

$$w'(m) = w_0(m) - w_l(m) - w'_r(m)$$

where $w_0(m)$ and $w_l(m)$ remain the same, but

$$w_r(m) = \sum_{\substack{(v_1,v_2):t\notin E \\ (m(v1),m(v2)):t\in E'}} \overline{g}(v_1) + \overline{g}(v_2)$$

Furthermore, there seems no way to guarantee that the triangular inequality holds for any output of the phase 1. Despite that, it seems quite reasonable to assume that this property holds for any such guess. In fact, we believe finding a **real** guess in which this does not hold is unlikely.

Another question which may arise here is about complexity of the problem. Supposing that it is efficient, one can come to an efficient way for solving the graph isomorphism problem; given a pair of (un-typed) graphs (not embedded on a metric space), assign a fixed type t to all of the edges, embed them on a metric space in which the distance of any pair of points is 1, and run our algorithm on them - **in an efficient time**. The heaviest matchings can be efficiently checked for being an isomorphism, because one can remove the types and the metric

[5] For a note on how to prevent this formula to approach to infinity, please refer to the commentary.

space backbone. It is easy to verify that there is a isomorphism between the original graphs iff the mapping with the biggest weight is a isomorphism between them.

In this paper, we assume that for considering all the possible matchings, one iterates through matchings until making sure they are finished. This means the algorithm iterates exponential times. Nevertheless, considering all the possible matchings is not needed. As Papadimitriou and Steigiltz show in [10], there exist heuristics for dealing with this in a P time. For the moment, however, we do not consider those heuristics. Despite that, we are not about to leave this problem in its general form; We believe that the following ontology-matching-specific heuristics can decrease the runtime. For each of these heuristics, a rational is also presented.

C. Heuristics for Decreasing the Runtime

All the heuristics presented here are based on the types of edges. The following list shows the whole idea: (Let's call this list the recipes for discard and contraction.) In this list, for the first and third item, we change the initial graph via contraction along its certain parts, then apply our refinement method to the resulting reduced graph, and finally transform the graph back to what it has originally been. Having done this, we consider completing the proposed mappings by moving back to consideration the neglected parts during the period when the graph was in its contracted form. We will call this restore of contracted vertices the *expansion phase*.

- *IS-A (rdfs:subClassOf)*: Contract all the paths into a pair of vertices between which there is an edge of type IS-A. The source of this edge will the source of the original path, whilst the destination will be a new vertex, similarity of which is the maximum of the similarities of the original path excluding the source. At the expansion time, consider this problem as an independent matching problem, but with the explanation stated below.
- *Disjoint (owl:disjointWith)*: If the difference between the distances of a concept in one ontology from a couple of disjoint concepts in another is above a certain threshold, remove the possibility of mapping the first concept with the one in the couple which is farther.
- *Equivalence (owl:equivalentClass)*: Contract all such vertices into one representing the whole group. Assign the maximum similarity of group to this new node. On expansion, there is no difference between different choices for matching between the two graphs.
- *owl:functionalProperty*: Functional properties should be mapped to functional properties, so, discard all the mappings in which this does not hold.

As far as the authors understand, all of the above heuristics should seem rational except the first one. To have an intuition on the contraction, one can consider it like Query Expansion in the Information Retrieval [14] terminology. The expansion however is a little tricky. There is a fine observation which should be made on an IS-A paths. Consider Fig. 4.I, in which after expansion, it is chosen to map a to a', and b to b'. Here, there remains no choice for c! Now, consider fig. 4.II, in which

Property	NoU	PI+	PI-
owl:incompatiblewith	0	0	0
owl:alldifferent	13	0.01	0.01
owl:differentfrom	13	0.01	0.01
rdfs:datatype	11	0	0.01
owl:symmetricproperty	27	0.01	0.02
owl:sameas	43	0.02	0.03
owl:equivalentproperty	70	0.03	0.05
owl:inversefunctionalproperty	100	0.04	0.08
owl:thing	233	0.09	0.18
owl:transitiveproperty	266	0.11	0.21
owl:oneof	313	0.12	0.24
owl:maxcardinality	807	0.32	0.63
owl:inverseof	932	0.37	0.73
owl:mincardinality	1315	0.52	1.02
owl:unionof	1629	0.65	1.27
owl:cardinality	2416	0.96	1.88
owl:allvaluesfrom	2841	1.12	2.21
rdfs:subpropertyof	2893	1.15	2.25
owl:equivalentclass	4836	1.91	3.76
owl:functionalproperty	7625	3.02	5.93
owl:disjointwith	7892	3.12	6.14
rdfs:domain	8476	3.36	6.59
owl:intersectionof	9482	3.75	7.38
owl:somevaluesfrom	22874	9.06	17.79
owl:restriction	53440	21.16	41.57
rdfs:subclassof	124005	49.1	-
Sum	252552	-	-
Sum without subclass of	128547	-	-

TABLE I
FREQUENCY OF OWL (AND RDF) PROPERTIES

a is mapped to a'. Note that because $b\ IS-A(n)\ a$, and $b'\ IS-A(n)\ a'$, it is not wise to map b to a', and there remains no choice for either of b and c. With this schema in mind, a solution to the expansion will become trivial, and the complexity of which will definitely be too small - say $O(n)$! Therefore, we will not delve into details of this.

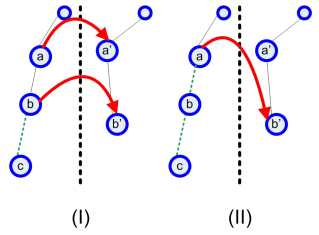

Fig. 4. Notes on Expansion Phase of IS-A

A question which may arise here is that "Why are there only a few properties chosen amongst the set of all *OWL* and *RDF* properties?" The reason behind this choice is a survey we have performed on a set of 545 ontologies. The following Table I shows the results of this survey (where *NoU* = Numb of Usage, *PI+* = Percent of usage with *IS-A*, *PI-* = Percent of usage without *IS-A*). As you can see here, amongst all the properties taking part in this survey, only the ones we have chosen heuristics for have a considerable percent of usage. This means that our choice should be enough here – which is the best estimate so far on how appropriate our work is.

-	n	o	p
b	0.9	0.1	0.4
c	0.6	0.7	0.1
d	0.4	0.5	0.6
e	0.4	0.5	0.4

-	(p,n)	(o)
(b,d)	0.9	0.5
(b,e)	0.9	0.5
(c)	0.6	0.1

TABLE II

DISTANCE OF NODES BEFORE AND AFTER OF CONTRACTION

1	(b,d), (p,n)	(b,e) , (o)	0	1.4	1.4	-2.8
2	(b,e), (p,n)	(b,d), (o)	0	1.4	1.4	-2.8
3	(b,d), (p,n)	(c), (o)	0	1.0	1.0	-1.4
4	(c), (p,n)	(b,d), (o)	$\frac{1}{0.6}+\frac{1}{0.7}$	0	0	3
5	(b,e), (p,n)	(c), (o)	0	1.0	1.0	-2.0
6	(c), (p,n)	(b,e),(o)	$\frac{1}{0.6}+\frac{1}{0.7}$	0	0	3

TABLE III

THE EXAMPLE AFTER CONTRACTION

D. Our entire Mechanism

Putting all the above ideas together is not hard. To make it short, here is a pseudo-code:

1) Input O and O'.
2) Apply a Threshold Based Refinement on O and O'.
3) Apply the recipes for Discard and Contraction on O, O'; call the resulting ontologies O_1 and O'_1.
4) Weight all the remained possible mappings from O_1 to O'_1.
5) Expand back the contracted parts of O_1 and O'_1.
6) Output the mappings along with their weights.

As an example of how this works, consider Fig. 5:

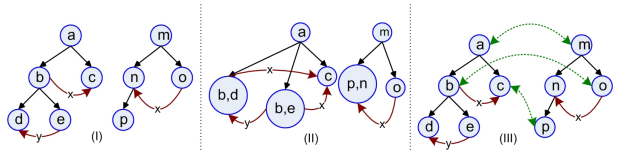

Fig. 5. The Example, I- Before, II- After Contraction, III- Final Mapping.

In this figure, you can see two ontologies O and O', and II-left d showing the distance between concepts across them.

Suppose that the d above is the one after the second step. By the end of the third step, Fig. 5-II will be the result, and d will then change to Table II-right.

Choosing $f(x) = x + 0.1$ and $g(x) = x$, Table III will be the outcome of step 4. This shows that, so far, either of mappings 4 or 6 can be chosen as an ideal. This means that the problem is now reduced to two simpler subproblems: In the first, one should decide on mapping either of p and n to c, and, in the second, on choosing between b and d to be mapped to o. Considering the individual distances between vertices, one can easily choose to map b to o, and c to p. The final matching, therefore, will be what is depicted in Fig. 5-III.

V. FUTURE WORKS

As told in the text, in the current work, we suppose that, in the step 4 above, we iterate through all the possible mappings from O_1 to O'_1, which will be an exponential piece of job. Therefore, one possibility for future works is considering the works of Papadimitriou and Steigiltz [10], and try to inject them into this problem to come to a polynomial algorithm. Another possibility could be considering other works in which Graph Theory and Metric Spaces are considered together, and find new ideas for further reduction of the size of this method. One can consider [15] for example. Given that it is common to use Domain Theory [13] for evaluating the semantics of programming languages [16], we believe that there is a vast room for injecting those ideas in the realm of ontology mapping, especially in better adjustment of the partial order we were speaking about in this paper.

VI. ACKNOWLEDGMENTS

Many thanks to Prof. Richard M. Wilson for his kind comments on typed graphs, Dr. Mohammad Mahdian for his notes on the heuristics on isomorphism, and Taowei David Wang for his fertile data set containing the ontologies we have used here. Furthermore, we would like to give a thank to all the people at the Ontology and DL mailing list who helped.

REFERENCES

[1] P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris, "Specification of a common framework for characterizing alignment," Knowledge web NoE, deliverable 2.2.1, 2004.

[2] M. Ehrig and Y. Sure, "Ontology mapping – an integrated approach," in *Proc. 1st European Semantic Web Symposium (ESWS)*, ser. Lecture Notes in Computer Science, C. Bussler, J. Davis, D. Fensel, and R. Studer, Eds., vol. 3053. Hersounisous (GR): Springer Verlag, May 2004, pp. 76–91.

[3] P. Valtchev and J. Euzenat, "Dissimilarity measure for collections of objects and values," in *Proc. 2nd Symposium on Intelligent Data Analysis (IDA)*, P. C. X. Liu and M. Berthold, Eds., vol. 1280, 1997, pp. 259–272.

[4] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: a versatile graph matching algorithm," in *Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US)*, 2002, pp. 117–128.

[5] A. Gibbons, *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[6] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.

[7] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *Journal on Data Semantics*, vol. IV, 2005.

[8] *Webster's New World College Dictionary*, 4th ed. New York: Macmillan, 1998.

[9] W. A. Wilson, "On quasi-metric spaces," *American Journal of Mathematics*, vol. 43, pp. 675–684, 1931.

[10] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, 1998.

[11] G. Polya, R. E. Tarjan, and D. R. Woods, *Notes on Introductory Combinatorics*, ser. Progress in Computer Science. Boston/Basel/Stuttgart: Birkhaeuser, 1983, vol. 4.

[12] D. West, *Introduction to Graph Theory (2nd Edition)*. Upper Saddle River): (Prenctice Hall, 2001.

[13] S. Abramsky, "Domain Theory in Logical Form," 1987.

[14] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.

[15] B. Xiao, H. Yu, and E. Hancock, "Graph matching using spectral embedding and semidefinite programming," in *Proceedings of the 15th British Machine Vision Conference*, 2004.

[16] R. Tennent, "The denotational semantics of programming languages," *Communications of the ACM*, vol. 19, p. 437, 1976.