# OMeGA: Ontology Matching enhanced by Genetic Algorithm

Mehrnoush Shamsfard
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran,
m-shams@sbu.ac.ir

Behzad Helli
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran,
Behzad_helli@yahoo.com

Samira Babalou
Department of Computer Engineering, Faculty of Engineering, University of Science and Culture, Tehran, Iran,
Samira_Babalou@yahoo.com

*Abstract*—**In this paper, we propose a new ontology matching approach, OMeGA, based on genetic algorithms applied on the graph structure of ontologies. Our approach finds the linguistic-structural similarities between concepts in two ontologies. It introduces new fitness functions and new criteria for categorizing test cases into four categories. Our approach does not need any extra information or resource with exception to the ontology itself. Experimental results on applying OMeGA on defined cases show higher performance compared to existing method.**

*Index Terms*—**Ontology matching, genetic algorithms, graph theory.**

## I. Introduction

It is well known that ontologies play a major role in computer science and engineering and many related fields such as social network [1], geographic information systems [2], e-commerce [3], data warehousing [4] ,natural language processing [5], multi-agent systems [6], information retrieval [7]. Many systems and applications employ various ontologies on various domains. Ontology mapping, integration, alignment and matching are known to be among ontology engineering activities, which enable interoperability among these information and knowledge based systems.

Ontology matching refers to the process of finding the correspondence between semantically related elements in two different ontologies. It allows the knowledge and data expressed in the matched ontologies to interoperate [8] and can be used for various tasks such as ontology evaluation, question answering, web service discovery, navigation on the semantic web and so on. In this paper, we introduce a new method for ontology matching that exploits genetic algorithms.

There are various ontology matching approaches focusing on different elements and features of ontologies. They may be applied in different levels from the shallowest one which corresponds to lexical similarities between labels to the deepest one which corresponds to matching the semantics of the elements.

Ontology matching systems usually employ a similarity discovery approach to calculate a similarity measure between elements of two ontologies and compare it to a standard in order to find corresponding elements. A matching system may use one or more simple matchers including name matcher, description matcher, property/restriction matcher, structure matcher and semantic matcher to calculate similarities.

Name matchers (and also description matchers) look for linguistic similarity of two strings of the concepts' labels (or descriptions). Similarity between labels can be calculated by string matching methods such as longest common substring or minimum edit distance. In some systems linguistic information about the morphology or meaning of words can help the system to find equivalent or similar labels.

Structure matchers focus on the structure of ontologies, such as the topology of the graph or the shape of inclusion hierarchy. In these matchers concepts are similar if they occur in similar structures, which is defined by have similar relations to similar concepts or similar attributes with similar values. Studying the similarities between the restrictions on the attribute-values by property matchers may enrich the second case too. In this paper, we introduce a genetic based structure matcher, which employs a name matcher to enhance the results.

## II. Related work

There are different approaches used in ontology alignment, matching, merging and integration. Using artificial neural networks (as in CIDER-CL [9]), Markov logic (as in CODI [10]), parallelism (as in GOMMA [11]) and machine learning techniques (as in YAM++ [12]) are some of the examples.

Exploiting genetic algorithms to align ontologies is another topic which has gained attention in recent years. In GAOM (Genetic Algorithm based Ontology Matching) [13], Wang and colleagues used genetic algorithms as their main approach. Their fitness function was built up using a set of correctly matched nodes and a set of non-matched nodes. However, the two sets created a weakness in their method because there is no way to determine a correct match. Hence they counted the matched items, versus the unmatched ones.

In another research paper, Jorge Martinez-Gil et al. presented GOAL (Genetics for Ontology Alignments) [14]. They used four different fitness values, each for improving an attribute of the alignment. The values are precision, recall, f-measure and fallout. They used these values as different fitness values, but the problem is that these values are unable to be precise when the goal alignment lacks definition or information.

As another past example, Ginsca and Iftene [15] used genetic algorithms to optimize the similarity aggregation step in ontology alignment. They first calculated the basic similarity measures such as the syntactic, taxonomy and semantic measures, then optimized the aggregation of these measures using genetic algorithms. Using the same method, Naya and colleagues [16] combined multiple similarity measures with genetic algorithms. The drawback of [15, 16] is they required a priori knowledge about ontologies under alignment in order to select the most suitable set of the weights. The approach of [15] focuses on optimizing the whole similarity aggregation step as a single unit, including the threshold value in the chromosome.

Acampora et al. [17] used a combination of genetic algorithms and a hill climbing search to optimize the similarity aggregation in the process of ontology matching. Their approach simultaneously optimized both the combination of weights and the threshold value used to perform the cut operation. They optimized the whole similarity aggregation phase by including the threshold value as a part of the chromosome structure.

In this paper, a genetic based matching is presented that is based on the similarity of the graph structures. Ontology graphs need a great amount of process and space to find the best match, so we devised a system to solve this problem. We approached this by defining our system using the optimum data structures and base algorithms that could be made. The other feature of this method is its independency to any extra information or resource. Our approach gets good results by working on just the ontology itself and this is its advantage compared to some other structured based algorithms such as [18] in which the ontology hierarchy should be populated with properly classified text documents.

In the next section, our matching method is presented. For this purpose, at first, the genome and the generated fitness value are defined and then the methods of mutation and cross-over are shown. The third section is assigned to categorize various ontologies that should be matched. Section four shows the test bench we generated and used to evaluate the response of the proposed system to different cases. The last section discusses the conclusions.

## III. The Proposed Method

Matching Ontologies becomes complex when there is no additional information such as document frequency or meta-data about the two ontologies that are going to be matched. In this paper's proposed method, the only data that the system works on is the ontology itself and, as explained in the rest of the paper, the matching result has a better chance if the ontologies have more complex structures.

The proposed method is a genetic based one. As the system's process iterations are rather high, it was very important to reduce the complexity of the methods used in each of the iterations.

### *A. The Genome*

Each Genome represents a possible and valid matching between the two ontologies. We call a matching 'valid', if after applying matching and generating the result ontology; its structure is a valid ontology structure (e.g. it has no "Is-a" or "part of" loops). We call a matching 'a possible one', if each node in one graph, is matched to at most one node in the other graph. In ontology mapping in some cases, it may be possible to match one node to two other nodes, but to make the genome simpler; we only allow each node to be matched to at most one node.

In this context, we introduce each ontology by graph A and graph B. Each graph represents the structure of the ontology. In order to have such properties, we define the Genome, as a 2D matrix of dimensions of $N \times M$ as $Mat$, where $N$ is the number of nodes in graph A, $M$ is the number of nodes in graph B and each element in $Mat$ such as $Mat(i,j)$ shows if node $i$ of graph A is matched to node $j$ of graph B.

To check the validity of the structure, we have to find the probable loops over the relations which are without loops (such as "Is-a" and "part of"). To facilitate this, a graph was generated from the genome and then using the DFS algorithm the loops were searched for in the graph. The DFS algorithm has the complexity of $O(E)$, where E is the number of edges. A type of graph that must not have any loop is a Directed Acyclic Graph (DAG). The complexity of a validity check of such a graph is $O(EV)$, which calculates a DFS route from each node and checks if that node is in a loop. If the inheritance of the objects is set to mono-parent inheritance, which makes the graph form a tree and perform the validity check in a tree of $O(E)$ where $E$ and $V$ are the number of edges and vertices of the resulted graph, respectively.

The possibility check requires that no two nodes being mapped to the same node. To check the possibility of the genome, there must be no two true elements in Mat where they are in the same row or same column. The best method to do so is to flag each row or column used in the Genome, if a row or column is flagged more than once, that means the node that is represented by that row or column is matched to more than one node. Using such an algorithm, the complexity order of the possibility check will be $O(V)$.

If the whole matrix was stored, the space complexity would be $O(V^2)$. But such a genome, that has at most one true element in each row and column, is very sparse, so instead of storing the entire matrix, the Sparse-Matrix structure was used, which reduces the space complexity to $O(V)$).

### *B. The Fitness Function*

The evaluation of the fitness function involves the calculation and combination of four types of values, including:

- Node Positive (NP) value – each two nodes that are matched - needs a calculation of their similarity.
- Edge Positive (EP) value – each two edges in the graphs that are matched together - should improve the fitness.
- Edge Low Negative (ELN) value – any edge in one graph where both its nodes are matched to two nodes in the other graph, but the corresponding edge does not exist - must reduce the fitness value.
- Edge High Negative (EHN) value – any edge in one graph where both its nodes are matched to two nodes in the other graph, but the corresponding edge is reversed - must also reduce the fitness value. In this case, the difference is the value must be reduced with greater effect.

To generate the NP value, the string similarity method is used. The Longest Common Subsequence (LCS) algorithm is used to generate a similarity measure between the two labels. The value is generated as Eq. (1) and Eq. (2).

$$np(G_1:i,G_2:j)=\frac{Length(LCS((G_1:lalel_i,G_2:label_j)))}{\max(Length(G_1:label_i),Length(G_2:label_j))} \quad (1)$$

$$NP(G_1,G_2,Mat)=\left(\sum_{i,j} np(G_1:i,G_2:j)\middle|Mat(i,j)=1\right) \quad (2)$$

Because in many cases there is no similarity between the two labels, the proposed system was also tested without using the NP value. In that case, the system only tries to match the best patterns of edges to each other.

To generate the EP value for each edge that has been correctly matched to another edge in the graph needed the calculation of a defined value ($T_{EP}$). The value of $T_{EP}$ results in a variation of the fitness, so the effect of different values of $T_{EP}$ were evaluated. Eq. (3) shows the resultant EP value.

$$EP(G_1,G_2,Mat)= \quad (3)$$
$$\left(\sum_{ai,aj,bi,bj} T_{EP}\middle|G_1:E_{(ai,bi)}\ \&\ G_2:E_{(aj,bj)}\ \&\ Mat(ai,aj)\ \&\ Mat(bi,bj)\right)$$

Generating the ELN and EHN values has a similar method to the EP value. In these cases, we check if the edge is mismatched. Eq. (4) and Eq. (5) respectively show the amount of ELN and EHN.

$$ELN(G_1,G_2,Mat)=\left(\sum_{ai,aj,bi,bj} T_{ELN}\middle|\begin{matrix}G_1:E_{(ai,bi)}\ \&\ G_2:E_{(aj,bj)}\ \&\\ (Mat(ai,aj)+Mat(bi,bj)=1)\end{matrix}\right) \quad (4)$$

$$EHN(G_1,G_2,Mat)=\left(\sum_{ai,aj,bi,bj} T_{EHN}\middle|\begin{matrix}G_1:E_{(ai,bi)}\ \&\ G_2:E_{(bj,aj)}\ \&\\ Mat(ai,aj)\ \&\ Mat(bi,bj)\end{matrix}\right) \quad (5)$$

The ELN value, counts the edges that have no corresponding match in the other graph. And the EHN value, counts the edges that have a corresponding match in a reverse form.

### C. *Crossover and Mutation*

The OMeGA algorithm can be described as follows:

(1) Having genome A and genome B, genome C is generated as the exact common genes that they have (common matches).

(2) For $N \times M$ times, a slot of the gene matrix is randomly selected.

  a. If the gene was equal to one, for a probability of destruction ($P_d$), the value becomes zero.

  b. If the gene is equal to zero, for a probability of construction ($P_c$) the value is changed to one; as long as the resulted genome is still valid and possible.

The above algorithm calculates every two genomes in the population. Also the two best genomes (those with highest fitness values) of the last population are added to this new set. Then the fitness of all the new genomes are calculated and the best size of population, denoted k, genomes are selected to be in the next generation. Fig. 1. shows the flowchart of OMeGA.

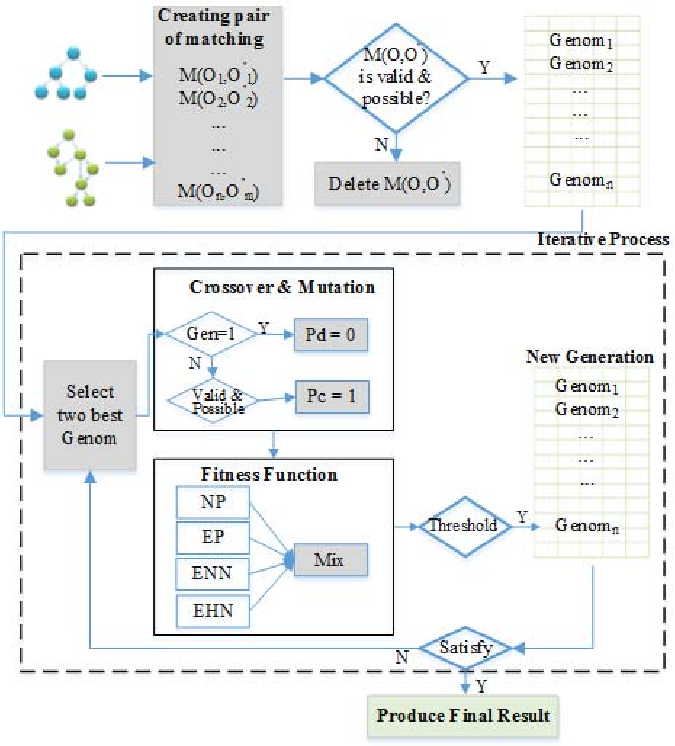
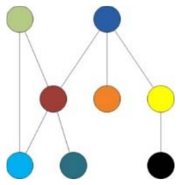


Fig. 1. OMeGA flow diagram.

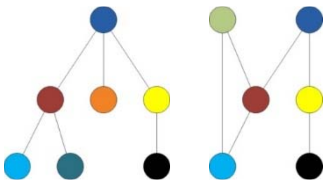
## IV. Different Cases of Matching

Different cases of matching may occur, depending on the similarity of the two ontologies. In order to classify these cases, two similarity measures are defined.

First is the similarity of the structures of the two ontologies. The two structures may be semi-identical or partially similar. We call two structures, semi-identical structures, when the basic structure of them is the same, but they may have or lack some of the nodes from the basic form. A sample of such ontologies is shown in Fig. 2. But if the basic form is not the



(a) Base structure



(b) 1st Ontology 2nd Ontology

Fig. 2. A sample of semi-identical structures

same, we call them partially similar structures. Such difference may occur in different languages or different views of the same domain.

The second aspect is the similarity of the labels of the nodes. Again the two ontologies may have semi-identical labels or not. Semi-identical labels are in form of:

$label_1 =< pre_1 > l < post_1 >$ and

$label_2 =< pre_2 > l < post_2 >$

which means, they have an identical substring. (It is possible to have more than one pair of identical substrings in the labels). Or they may have no similarities, which may occur when the labels are from different natural languages or while an object has more than two notations in the language. A sample of these situations is shown in TABLE I. .

TABLE I. SAMPLE OF SEMI-IDENTICAL AND NON SEMI-IDENTICAL LABELS

| 1st Label | 2nd Label | Type of similarity |
|---|---|---|
| **Humans** | Human | Semi-identical |
| **Humans** | Human-form | Semi-identical |
| **Humans** | People | Non Semi-identical |
| **Humans** | Man-kind | Non Semi-identical |
| **Humans** | انسان[1] | Non Semi-identical |

In the rest of this section we discuss how OMeGA matcher deals with various cases which occur according to the above two similarity measures.

### A. *Semi-Identical Structures, Semi-Identical Labels*

In this case, the two ontologies have the same basic structure and the labels are semi-identical, the difference is that there are some nodes in one ontology that are not present in the other ontology and vice versa. An example of such two ontologies is shown in Fig. 3. .

[1] Means Human in Persian and Arabic

These types of ontologies are usually created when a big domain is divided into smaller domains whose ontologies are developed by the members of a research group. . The resulted small ontologies are represented in such a form. In order to match these ontologies, the basic form of the system should be used.

### B. *Semi-identical Structures, non semi-identical Labels*

In this case, the basic structure of the ontologies is the same, but the labeling is different. Fig. 3. can be a sample of such ontologies if the labeling in one of the ontologies was done in another language. These types of ontologies are more common than the others. There could be cases where some labels are semi-identical but there are also labels that exist with no similarity between them.

In order to match those two types of ontologies, it's better to calculate and use the NP value; but if its value is near one, there a greater emphasis is required opposed to when there is almost no similarity between labels.

### C. *Partially similar Structure, semi-identical Labels*

In this case, the two ontologies must be matched mostly by their labels. For example in Fig. 4. , the "human" node has gotten to a place that is both child of "mammal" and "living thing".

In this case, we should emphasize the label's similarity effect more than the basic form of the system. So it can match the ontologies more on their node values.

### D. *Partially similar Structure, non semi-identical Labels*

In this case, even a human agent can't be absolute in generating the best match, especially when the ontologies are not in same language. The only information that is available in this situation is the partial similarity of the structures. So any system should try to match the best sub-ontologies to each other. A sample of such a match is shown in Fig. 4. , when one of the ontologies is in another language. In such a case the human node is not recognizable.

The proposed system can only be used to generate small local suggestions and can't be absolute on the whole ontology. This condition is not regular and is not mentioned in standard test benches.

## V. TEST BENCHES & EXPERIMENTAL RESULTS

In order to test the system in different situations, two test benches were generated. The first test set is taken from [19]. This dataset consists of four groups of ontologies (animals, russia, tourism, sport) and each group has two OWL ontologies and reference matching results. For the second test bench we developed an application to generate two ontologies that could get mapped to each other. To accomplish this, first a random ontology with N vertices was made, then the relevant ontology was duplicated, which involved a random elimination of some nodes of the original ontology and some of the names were disfigured. Thereafter, exist two ontologies that embed a partial match. Using such an ontology generator, different ontologies with different nodes and relation numbers can be generated, so

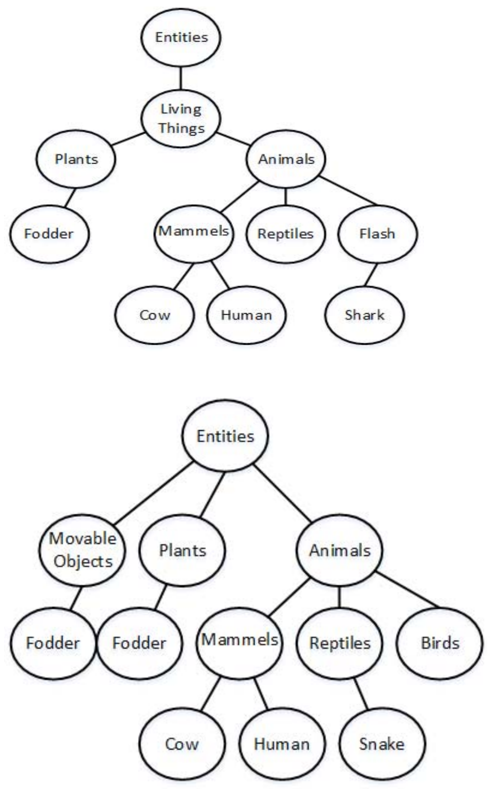


Fig. 3. Sample of two Ontologies with semi-identical structure and labels

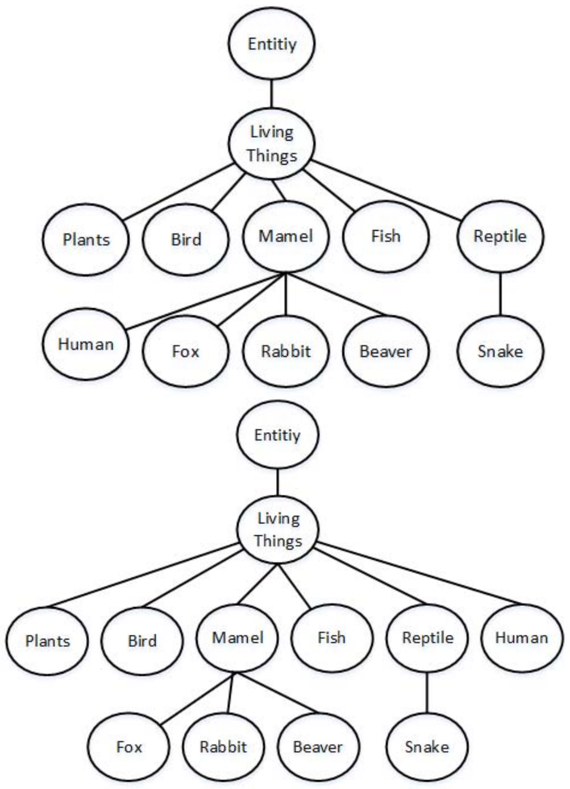


Fig. 4. The difference in structure is occurred because of the difference in beliefs. One expert believes humans are mammals, but the other one categorizes it under living things

the tests are more controllable by defining different number of nodes and relations.

The FOAM test bench was gathered from ontologies that had semi-identical structures, in some nodes the labeling was exactly the same and in some other cases the labeling had no similarities. So, the second form of the application was used on them. The results of the FOAM ontologies are shown in TABLE II. .

We use standard information retrieval metrics to assess the results of our tests that are shown in Eq. (6), Eq. (7), and Eq. (8).

$$Precision = \frac{number\ of\ correct\ found\ alignments}{number\ of\ found\ alignments} \quad (6)$$

$$Recall = \frac{number\ of\ correct\ found\ alignments}{number\ of\ exixting\ alignments} \quad (7)$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (8)$$

TABLE II. RESULTS OF APPLYING OMEGA ON FAOM DATASET

| **Ontology name** | | **Russia** | **Tourism** | **sport** | **Animals** |
|---|---|---|---|---|---|
| **Pair type** | | 1 | 2 | 2 | 2 |
| **OMeGA** | precision | 0.997 | 0.95 | 0.99 | 1.00 |
| | Recall | 0.98 | 0.96 | 0.99 | 1.00 |
| | F-measure | 0.99 | 0.95 | 0.99 | 1.00 |
| **[20]** | Precision | 0.98 | 0.96 | 0.90 | 1.00 |
| | recall | 0.69 | 0.89 | 0.93 | 0.78 |
| | F-measure | 0.81 | 0.92 | 0.92 | 0.88 |

As it can be seen in TABLE II. , the results show the superiority of our approach over the approach proposed by Shen et al. [20].

As the ontologies of the first have the same structure, in order to complete the tests we developed a program to generate a pair ontology with differences in four scenarios.

1) Generating Type 1 ontologies.
   a) A random ontology with N nodes and M relationships is generated.
   b) The ontology is duplicated
   c) For K times, a random node is selected from each ontology
      i) It might be deleted
      ii) Its name might be transformed in a way where it pertains similarities to the prior name (this operation is limited to two iterations on one node)
2) Generating Type 2 ontologies.
   a) A random ontology with N nodes and M relationships is generated.
   b) The ontology is duplicated
   c) For K times, a random node is selected from each ontology
      i) It might be deleted
      ii) Its name might be transformed randomly
3) Generating Type 3 ontologies.

a) A random ontology with N nodes and M relationships is generated.
b) The ontology is duplicated
c) For K1 times, a random node is selected from each ontology
   i) Its name might be transformed in a way that it is still similar to the last name (this operation is limited to two iterations on one node)
d) For K2 times, a random edge is selected from each ontology
   i) It may be deleted
   ii) It may be randomly put to another place

4) Generating Type 4 ontologies.
   a) A random ontology with N nodes and M relationships is generated.
   b) The ontology is duplicated
   c) For K1 times, a random node is selected from each ontology
      i) Its name might be transformed randomly
   d) For K2 times, a random edge is selected from each ontology
      i) It may be deleted.
      ii) It may be randomly put to another place.

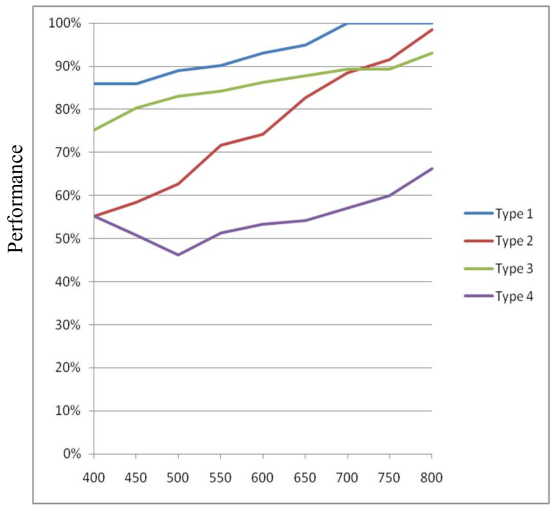
Such types of ontology pairs were generated while the number of edges was closely observed, so in steps of 50, the system was executed over the remaining ontologies.

As Fig. 5. shows, the proposed system has a predictable response to type-1 ontology pairs. Its performance rises when there are more edges to be matched and it also gets faster. The answer is attained faster when the conditions of the nodes are more unique in such cases when there are more edges to be matched. In case of type-2 ontologies, its performance increases by increasing the number of edges; however, the number of iterations needed to get to the best result at first increases and then decreases. It increases because at first the similarity of nodes and edges are found in contradiction, so it takes longer to decide, but after a while enough edges exist so, the best match becomes easier to identify. In type-3 ontology matching, the performance increases slightly in response to an increase in the number of edges, but because the structures are in contrast with each other, more iterations are needed to reach a stable solution. (It's important to mention that at least half of the similar structures are not destroyed in the random ontology pair generator). In type-4 situation, the behavior of the system is similar to the type-3 case, with lower performance.

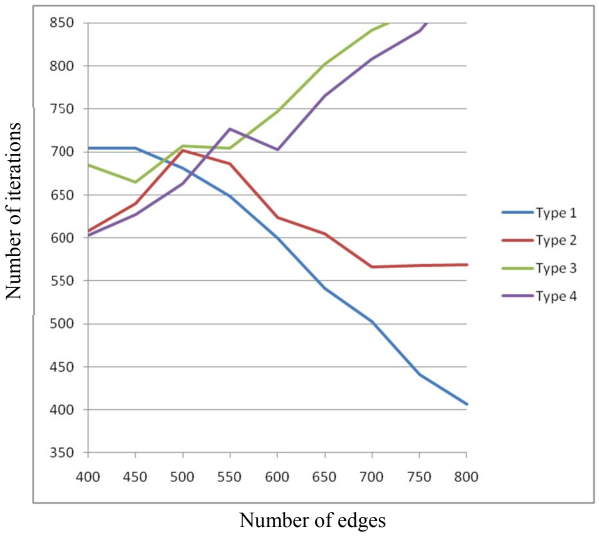
## VI. Conclusion

The proposed system, in spite of other ontology matcher systems, is only based on the structure. This paper has shown it to be a powerful system, due to its independence of any extra information on the ontologies - like most ontology matchers based on f-measure that are generated using extra documents.

To reduce the computational complexity at each iteration, only the valid and possible matchings are used to define genomes. Besides, using new structures reduces the space complexity from $O(V^2)$ to $O(V)$. In this paper, four new fitness functions have been suggested. Also, according to two defined criteria (label versus structure and semi versus partial similarity) four different cases are created and studied. Results of this study show that OMeGA algorithm, can find linguistic and structural similarities with high performance.



(a)



(b)

Fig. 5. (a) the performance of the system, according to the number of relations. (b) the number of iterations done by the system.

## References

[1] R. Lecocq, E. Martineau, and M. F. Caropreso, "An Ontology-based Social Network Analysis Prototype," in *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2013 IEEE International Multi-Disciplinary Conference on*, 2013, pp. 149-154.

[2] F. T. Fonseca, M. J. Egenhofer, P. Agouris, and G. Câmara, "Using ontologies for integrated geographic

information systems," *Transactions in GIS,* vol. 6, pp. 231-257, 2002.

[3] A. Léger, G. Michel, P. Barrett, S. Gitton, A. Goméz-Pérez, A. Lehtola*, et al.*, "Ontology Domain Modeling Support for Multilingual Servicies in e-commerce: MKBEEM," 2000.

[4] J. Gitanjali, C. Ranichandra, M. Kuriakose, and R. Kuruba, "Ontology and Hyper Graph Based Dashboards in Data Warehousing Systems," *International Journal of Engineering & Technology (0975-4024),* vol. 6, 2014.

[5] J. A. Bateman, J. Hois, R. Ross, and T. Tenbrink, "A linguistic ontology of space for natural language processing," *Artificial intelligence,* vol. 174, pp. 1027-1071, 2010.

[6] M. Obitko and V. Marík, "Ontologies for multi-agent systems in manufacturing domain," in *Database and Expert Systems Applications, 2002. Proceedings. 13th International Workshop on*, 2002, pp. 597-602.

[7] H.-M. Müller, E. E. Kenny, and P. W. Sternberg, "Textpresso: an ontology-based information retrieval and extraction system for biological literature," *PLoS biology,* vol. 2, p. e309, 2004.

[8] P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt, "Ontology Matching," *The 7th International Semantic Web Conference,* 2008.

[9] J. Gracia and K. Asooja, "Monolingual and Cross-lingual Ontology Matching with CIDER-CL: evaluation report for OAEI 2013," *Ontology Matching,* p. 109, 2013.

[10] J. Huber, T. Sztyler, J. Noessner, and C. Meilicke, "CODI: Combinatorial optimization for data integration–results for OAEI 2011," *Ontology Matching,* p. 134, 2011.

[11] T. Kirsten, A. Gross, M. Hartung, and E. Rahm, "GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution," *J. Biomedical Semantics,* vol. 2, p. 6, 2011.

[12] D. Ngo and Z. Bellahsene, "YAM++: a multi-strategy based approach for ontology matching task," in *Knowledge Engineering and Knowledge Management*, ed: Springer, 2012, pp. 421-425.

[13] J. Wang, Z. Ding, and C. Jiang, "Gaom: Genetic algorithm based ontology matching," in *Services Computing, 2006. APSCC'06. IEEE Asia-Pacific Conference on*, 2006, pp. 617-620.

[14] J. Martinez-Gil, E. Alba, and J. F. Aldana-Montes, "Optimizing ontology alignments by using genetic algorithms," in *Proceedings of the workshop on nature based reasoning for the semantic Web. Karlsruhe, Germany*, 2008.

[15] G. Alexandru-Lucian and A. Iftene, "Using a genetic algorithm for optimizing the similarity aggregation step in the process of ontology alignment," in *Roedunet International Conference (RoEduNet), 2010 9th*, 2010, pp. 118-122.

[16] J. M. V. Naya, M. M. Romero, J. P. Loureiro, C. R. Munteanu, and A. P. Sierra, "Improving ontology alignment through genetic algorithms," *Soft computing methods for practical environment solutions: techniques and, studies,* pp. 240-259, 2010.

[17] G. Acampora, U. Kaymak, V. Loia, and A. Vitiello, "Hybridizing genetic algorithms and hill climbing for similarity aggregation in ontology matching," in *Computational Intelligence (UKCI), 2012 12th UK Workshop on*, 2012, pp. 1-6.

[18] K. Todorov and P. Geibel, "Ontology mapping via structural and instance-based similarity measures," in *The 7th International Semantic Web Conference*, 2008, p. 224.

[19] M. Ehrig and Y. Sure, "Foam–framework for ontology alignment and mapping results of the ontology alignment evaluation initiative," in *Integrating Ontologies Workshop Proceedings*, 2005.

[20] G. Shen, Z. Huang, X. Zhu, L. Wang, and G. Xiang, "Using Description Logics Reasoner for Ontology Matching," in *Intelligent Information Technology Application, Workshop on*, 2007, pp. 30-33.