information

Article

Automatically Specifying a Parallel Composition of Matchers in Ontology Matching Process by Using Genetic Algorithm

Marko Gulić [1,*], Boris Vrdoljak [2] and Marina Ptiček [2]

1 Faculty of Maritime Studies, University of Rijeka, Studentska ulica 2, 51000 Rijeka, Croatia
2 Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia; boris.vrdoljak@fer.hr (B.V.); marina.pticek@fer.hr (M.P.)
* Correspondence: marko.gulic@pfri.hr; Tel.: +385-51-338-411

Received: 25 April 2018; Accepted: 6 June 2018; Published: 7 June 2018

Abstract: Today, there is a rapid increase of the available data because of advances in information and communications technology. Therefore, many mutually heterogeneous data sources that describe the same domain of interest exist. To facilitate the integration of these heterogeneous data sources, an ontology can be used as it enriches the knowledge of a data source by giving a detailed description of entities and their mutual relations within the domain of interest. Ontology matching is a key issue in integrating heterogeneous data sources described by ontologies as it eases the management of data coming from various sources. The ontology matching system consists of several basic matchers. To determine high-quality correspondences between entities of compared ontologies, the matching results of these basic matchers should be aggregated by an aggregation method. In this paper, a new weighted aggregation method for parallel composition of basic matchers based on genetic algorithm is presented. The evaluation has confirmed a high quality of the new aggregation method as this method has improved the process of matching two ontologies by obtaining higher confidence values of correctly found correspondences and thus increasing the quality of matching results.

Keywords: ontology matching; weighted aggregation; parallel composition; basic matchers; genetic algorithm

1. Introduction

The amount of available data has increased rapidly due to advances in information and communications technology. Consequently, there exist many data sources that describe the same domain of interest. These data sources are usually designed independently of each other and thus are mutually heterogeneous. At some later point, such heterogeneous data sources describing the same domain of interest frequently need to be coupled. An ontology enriches the knowledge of a data source by giving a detailed description of entities and their mutual relations within the domain of interest. Therefore, the use of ontologies facilitates the integration of heterogeneous data sources that belong to the same domain [1]. In computer science and information science, ontology is a formal, explicit specification of a shared conceptualization [2]. Ontologies are expressed in an ontology language. One of the most popular languages is Web Ontology Language (OWL) [3] that is recommended by W3C organization. Ontology matching is the process of finding correspondences between entities of different ontologies [4]. Ontology matching is a key issue in the process of integrating heterogeneous data sources described by ontologies: if the ontology matching process is accomplished successfully, the management of data coming from different sources becomes much easier [4].

The need to perform ontology matching at the highest possible level of quality will be demonstrated through an example of data integration when searching for particular information, e.g., from multiple local data sources written in different data formats (relational databases, XML, RDF, etc.). Local sources can be described by local ontologies, which are integrated into a single common ontology through an ontology matching process, so that a user is able to make one single query that will then be translated and executed against each local source. Another example of successfully employing ontology matching is the search for information on the World Wide Web. An important task of the state-of-the-art search engines is to determine the user's intent, which is "hidden" within the query [5]. One of the solutions to the problem of inferring the intent of the query is to introduce ontologies as additional semantic components of webpages. Ontologies will support machines in grasping the understanding of the data and thus the user will be enabled with a more effective search of the Web, including an automated matching of different webpages.

When ontologies are relatively small, an expert can manually associate entities in an ontology to corresponding entities in another ontology. When ontologies are large (e.g., ontologies contain more than 10,000 entities), manual mapping is too time consuming [6]. Therefore, to automate the process of matching heterogeneous data sources by using ontologies and ontology matching, the ontology matching process needs to be automated. To automate the ontology matching process, an ontology matching system has to be developed [4]. It is necessary to automate those parts of matching process that are not comprehensible to ordinary (i.e., non-expert) users within the matching system. In this way, the users will be able to apply the system in the future without knowing the inner steps of the matching process.

Many ontology matching systems actually perform the complex matching task by applying several basic matchers, which determine the correspondences between particular entities (classes, properties, instances) of the ontologies submitted to the matching process [4]. A basic matcher computes correspondences between entities using information obtained from one or more components of the entire ontology. Consequently, the most common practice is to employ multiple basic matchers in order to utilize all information held within the ontologies [1]. The most common composition for integration of multiple matchers into the matching system is a parallel composition of matchers where basic matchers are executed independently of each other, while the aggregated correspondence for all basic matchers is computed afterwards. The potential solution for aggregation of correspondences obtained by basic matchers is a weighted aggregation. The weighted aggregation calculates the aggregated correspondences between each two entities of two different ontologies by considering the results from all basic matchers. A certain weighted factor is assigned to every basic matcher to determine the aggregated value of all correspondences. The problem is how to automatically determine the most adequate weighted factor for every basic matcher within the aggregation, because due to uniqueness of each ontology, the efficiency of any basic matcher depends on the implementation of the ontologies that are matching [7]. Possibly, the given ontology may lack some of the components, or there might be components to which particular basic matchers are not applicable. A basic matcher that uses those components for its correspondence computation process will produce a poor and questionable result therefore its weighted factor should be automatically set to a low value. However, this basic matcher that uses components which are completely or partially missing does not have to be completely excluded from the weighted aggregation process (weighted factor set to 0) because it can happen that certain component is partially implemented (e.g., the component label is defined for 1/10 of the total number of entities). Then the results of that matcher can help to obtain better aggregated matching results, but the system must recognize that the weighted factor of that matcher should be automatically decreased so that the results of this matcher would not be dominant in the final aggregated result. Thus, the focus of this paper is to automate the weighted aggregation of basic matchers in a parallel composition of basic matchers within the ontology matching system by using genetic algorithm to improve the efficiency and quality of the ontology matching process.

In this paper, we propose a DWGA (Determine Weights by Genetic Algorithm) method that uses genetic algorithm to determine the weighted factor of each basic matcher in the weighted aggregation of basic matchers. Genetic algorithm is nature-inspired metaheuristic based on the process of natural selection [8]. Nature-inspired metaheuristics mimic different natural systems and processes using mathematical models and algorithms [9]. As a metaheuristic may quickly give a sufficiently satisfactory solution to an optimization problem, it can be used to resolve the optimization problem of the determination of weighted factors of basic matchers in the weighted aggregation process. Thus, in this paper the weighted aggregation method, which uses genetic algorithm to resolve the problem of assigning the weighted factors, is proposed. The weighted aggregation with DGWA method consists of two steps: in the first step the DWGA method automatically determines the weighted factor of each basic matcher taking part in the aggregation, while in the second step the aggregated correspondences are computed using weighted aggregation parameters set in the previous step by DWGA.

Furthermore, the evaluation of the weighted aggregation with DWGA method was performed on the *Benchmark biblio ontology track* [10] by embedding this aggregation approach as a part of CroMatcher ontology matching system [11–13] which had one of the best results according to the *Benchmark biblio ontology track*. The *Benchmark biblio ontology track* is the largest test set (in the context of the largest number of pairs of ontologies that have to be matched) in the ontology matching system evaluation organized by Ontology Alignment Evaluation Initiative (OAEI) [14,15] and it is a reference point for evaluating the matching quality produced by an ontology matching system. A comparison between DWGA method and the newest version of Autoweight method (Autoweight+++—method for automatic determination of the weighted factors of basic matchers) which is implemented in the last version of CroMatcher system [11], was also made. The Autoweight method was included in the comparison because it outperforms all methods for automatically determining the weighted factors of basic matchers [6], therefore it is assumed that the comparison with this method is a good indicator of real quality of the new DGWA method. The comparison of the achieved results proves a high quality of the DGWA method, as the version of CroMatcher system with DGWA method produced approximately equal, but more reliable matching results (i.e., correct correspondences with a higher confidence value) than the version of CroMatcher system with Autoweight+++ method. The fact that more reliable matching results were obtained by the version of CroMatcher system with DGWA method represents significant progress in the matching process.

The paper is organized as follows. In Section 2, OWL language, basic terminology of ontology matching and matching system architecture with parallel composition of basic matchers are introduced. In Section 3 we discuss the related work. Section 4 explains in detail our DWGA method. In Section 5 the evaluation of our DWGA method is performed. Finally, the conclusion is given in Section 6.

2. Ontology Matching

2.1. OWL

As it has already been stated, an ontology is a formal explicit specification of a shared conceptualization of a domain [2]. The term conceptualization refers to an abstract model of a real-world area as understood by humans. Explicit specification refers to the explicit terms and definitions that describe the concepts and the relations of the abstract model. Each ontology is expressed by using an ontology language that gives definition of the concepts and relations in the real-world domain that must be described by the ontology. One of the most popular ontology languages is Web Ontology Language (OWL) [16], recommended by W3C (World Wide Web Consortium) [17] as an international standard for ontology representation. OAEI evaluation, which is the most trustworthy evaluation for ontology matching systems, holds a large collection of test ontologies that are expressed in OWL. Therefore, DWGA method is evaluated on the OAEI collection of test ontologies as a part of CroMatcher ontology matching system, which supports matching between two OWL ontologies.

2.2. Terminology

The basic terms of ontology matching, adopted from [1,4] are presented below.

Definition 1 (Ontology matching). *Ontology matching is the process of finding semantic relationships or correspondences between entities of different ontologies. Ontology matching is defined as function:*

$$A = f\ (O, O', p, r) \tag{1}$$

where alignment A is the matching result between ontologies O and O′, p is a set of parameters within the matching process, and r is a set of resources used in the matching process.

Definition 2 (Correspondence). *Correspondence is a probability value describing the degree of equivalence between entities of different ontologies. A correspondence is defined as:*

$$c\ (e_\mathrm{i}, e'_\mathrm{j}) = n \tag{2}$$

where e_i is an entity of the ontology O, e'_j is an entity of the ontology O′, and n is a real number from the interval [0, 1]. The higher the correspondence, the greater the similarity between two entities.

Definition 3 (Alignment). *Alignment A is a set of all correspondences c (e_i, e'_j) between entities e_i of ontology O and entities e'_j of ontology O′ that are determined while matching ontologies O and O′. Alignment is actually the output of an ontology matching process.*

Definition 4 (Highest correspondence). *A correspondence between the two entities $e_i \in O$ and $e'_j \in O'$ has the quality of being the highest correspondence if and only if it has higher confidence value (i.e., value of n in* ***Definition 2****) than any other correspondence of either e_i or e'_j with some other entity. A highest correspondence of e_i and e'_j will be shortly denoted as c_{max} (e_i, e'_j). A highest correspondence c_{max} (e_i, e'_j) is defined as:*

$$c_\mathrm{max}\ (e_\mathrm{i}, e'_\mathrm{j}) \equiv \{[\ c\ (e_\mathrm{i}, e'_\mathrm{j}) = \max\nolimits_\mathrm{k} \in O'\ c\ (e_\mathrm{i}, e'_\mathrm{k}))] \wedge [c\ (e_\mathrm{i}, e'_\mathrm{j}) = \max\nolimits_\mathrm{m} \in O\ c\ (e_\mathrm{m}, e'_\mathrm{j})]\} \tag{3}$$

The DGWA method uses the highest correspondence rule when evaluating possible solutions of determining weighted factors of basic matchers during the genetic algorithm process.

2.3. Ontology Matching System

Many ontology matching systems exist. Although each system is unique, the architecture of all ontology matching systems consists of similar components. In general, the matching process can be divided into three main types of components [1,4]: (1) basic matchers, (2) compositions of basic matchers and aggregation methods for basic matchers' results, and (3) methods for final alignment.

Each basic matcher uses information from one or more ontology components to determine correspondences between compared ontologies. Therefore, an ontology matching system generally consists of several basic matchers that use all available information from ontologies to improve the matching results.

Basic matchers should be interconnected to determine high-quality correspondences between entities of compared ontologies. As stated before, a parallel composition of basic matchers is the most common composition for integration of multiple matchers. As stated before, a parallel composition of basic matchers refers to a collection of basic matchers that are executed independently of each other. To aggregate the correspondences obtained by the basic matchers, an aggregation method must be implemented in the matching system. As stated before, the focus of this paper is on a new aggregation method for aggregating the obtained correspondences' results within the parallel composition of basic matchers based on genetic algorithm. Parallel composition of basic matchers and the aggregation of the results of correspondences obtained by basic matchers within the parallel composition is presented in Figure 1.

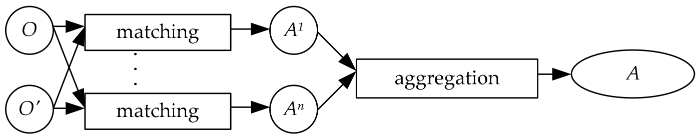
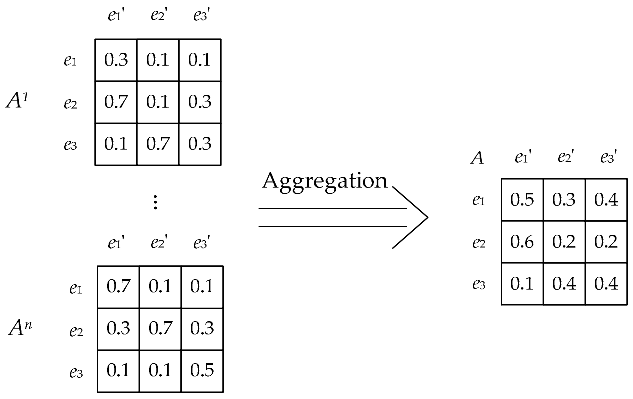


Figure 1. Parallel composition of basic matchers and the aggregation of the obtained matching results.

An alignment between two ontologies (A^n) is the result of the execution of a basic matcher. A^n contains the correspondences between all entities e_i and e'_j of compared ontologies (Figure 2). Considering the confidence values of correspondences within alignments $A^1 \ldots A^n$, an aggregation method produces the aggregated alignment by merging these confidence values of the correspondences between the same entities within $A^1 \ldots A^n$ according to defined procedure of aggregation method (Figure 2).



$$A^1 \quad \begin{array}{c|ccc} & e_1' & e_2' & e_3' \\ \hline e_1 & 0.3 & 0.1 & 0.1 \\ e_2 & 0.7 & 0.1 & 0.3 \\ e_3 & 0.1 & 0.7 & 0.3 \end{array}$$

$$\vdots$$

$$A^n \quad \begin{array}{c|ccc} & e_1' & e_2' & e_3' \\ \hline e_1 & 0.7 & 0.1 & 0.1 \\ e_2 & 0.3 & 0.7 & 0.3 \\ e_3 & 0.1 & 0.1 & 0.5 \end{array}$$

$$\xrightarrow{\text{Aggregation}}$$

$$\begin{array}{c|ccc} A & e_1' & e_2' & e_3' \\ \hline e_1 & 0.5 & 0.3 & 0.4 \\ e_2 & 0.6 & 0.2 & 0.2 \\ e_3 & 0.1 & 0.4 & 0.4 \end{array}$$

Figure 2. Aggregation of alignments obtained by basic matchers.

The most successful aggregation method in parallel composition of basic matchers is weighted aggregation method. As stated before, the weighted aggregation calculates the aggregated correspondence between each two entities of two different ontologies by considering the results from all basic matchers. A certain weighted factor is assigned to every basic matcher to determine the aggregated value of all correspondences. The weighted aggregation is defined as follows:

$$c\,(e_i, e'_j)^A = \sum w_n * c\,(e_i, e'_j)^{An}, \sum w_n = 1 \tag{4}$$

where $c\,(e_i, e'_j)^A$ is the aggregated correspondence between entities e_i of ontology O and entities e'_j of ontology O', $c\,(e_i, e'_j)^{An}$ is the correspondence between entities e_i of ontology O and entities e'_j of ontology O' produced by the nth basic matcher in parallel composition of basic matchers and w_n is the weighted factor of the nth basic matcher. As stated before, the main problem of weighted aggregation is how to automatically determine the most adequate weighted factor for every basic matcher within the aggregation. Well-defined weighted factors imply a quality aggregation and will certainly improve the efficiency and quality of the ontology matching process. Thus, in this paper a new weighted aggregation that employs genetic algorithm for automatic determination of weighted factors is presented.

Once the aggregated correspondences between all entities of the compared ontologies have been determined, a final alignment method must select the correct correspondences between entities of these ontologies (alignment A) to successfully finish the ontology matching process. The aggregation method also plays an important role in this final part of the matching process. If the aggregation method does not perform a quality aggregation, a final alignment method cannot successfully determine appropriate correspondences between entities of compared ontologies.

3. Related Work

As stated before, ontology matching systems usually consist of several basic matchers. Each basic matcher uses particular information within compared ontologies to determine correspondences between entities of these ontologies. According to [18], one of the greatest challenges in the ontology matching process is how to combine the results obtained by these basic matchers in order to make the matching between two ontologies successful. Certainly, the results of a basic matcher that achieved good matching results must be more involved in the aggregated result than the result of a basic matcher that did not achieved good results. For example, when matching two ontologies, a hierarchy of classes using the rdf:subClassOf tag is created within the first ontology, while it is not created in the second ontology. Consequently, the basic matcher based on the rdf:subClassOf tag for comparing a relation between two ontology classes will produce poor results. Hence, a quality aggregation method must automatically recognize good (or poor) results obtained by a particular matcher and increase (or decrease) the influence of this matcher to produce better aggregation results and maximize the efficiency of the ontology matching process. There are various approaches that propose different aggregation methods in parallel composition of basic matchers and they will be presented in this section. Also, the previous use of genetic algorithm in the ontology matching process will be presented.

The Coma system [19,20] is the ontology matching system based on parallel composition of basic matchers. This system is the predecessor to the most state-of-the-art systems. Four aggregation methods are implemented within this system: minimal value, maximal value, average and weighted aggregation. The methods minimal value and maximal value take the correspondence with minimal (maximal) value among the results of different matchers as the aggregated correspondence result for the two entities. In this way, only one maximal (minimal) correspondence value between two entities is taken into calculation of aggregation correspondence between these entities and this is the main limitation while using minimal and maximal value. The selection of the maximal (minimal) correspondence value between two entities obtained from basic matchers is not necessarily accurate. For example, considering the maximal value, the basic matcher Prefix, which checks whether a string is the prefix of another string, finds maximal similarity between elements Node and NodePair, although the NodePair element represents a pair of nodes and Node represents only a single node. The average aggregation method resolves the shortcomings of the first two methods by taking into calculation all values of correspondences between two compared entities. The shortcoming of the average method is the equal importance of all basic matchers while calculating the aggregation correspondence result between two entities, which is usually not the case in real world. Considering the implementation of currently compared ontologies, one basic matcher can determine better matching results than another matcher therefore it should increase the impact of this better matcher on the aggregated result in the matching process. To overcome this shortcoming of the average method, the weighted aggregation was introduced. This method evaluates differently the impact of every basic matcher while calculating the aggregation result considering the overall quality of results obtained by an individual matcher. This aggregation method is the most used method in the aggregation process. The biggest challenge of the weighted aggregation is how to determine the weighting factor of an individual basic matcher i.e., how to determine the quality of matching results that a certain basic matcher achieved. Here, the authors did not use the weighted aggregation method, as they did not want to make any assumptions about the importance of the basic matchers. In this paper, we propose a method based on genetic algorithm that automatically determines weighting factors of every basic matcher for weighted aggregation.

The YAM++ [21] system, which is one of the best ontology matching systems considering the results of OAEI evaluations, contains several basic matchers that are aggregated through the weighted aggregation. The authors do not explain how they set the weighting factors for individual matcher. Therefore, it is assumed that the authors use their experience from earlier testing of YAM++ system to determine weighting factors manually. Furthermore, a various number of matching systems [22–27] uses the weighted aggregation as aggregation method in the matching process, without explaining the procedure of determining weighted factors of basic matchers.

The IAMA system [28] consists of four basic matchers. The results of the first three basic matchers are aggregated using the weighted aggregation. The weighted factors of these three basic matchers have been heuristically determined based on the authors' experience (values of the factors are always the same) and have always the same values. We believe that the weighted factors of a particular basic matcher should be automatically determined considering the implementation of currently compared ontologies. For example, if the matcher that determines correspondence between entities by comparing entity labels has the highest weighting factor, and the labels are not implemented in one of the matched ontologies, the system will not achieve good results. The aggregated result of the first three basic matchers and the result of the fourth matcher are aggregated by maximal value aggregation method. The limitation of the maximal value has been explained before in this section.

The CIDER-CL system [29] uses a neural network [30] to determine aggregate correspondences. The biggest challenge when using a neural network, which is a machine learning method, is the appropriate set of training examples for learning how to solve an unknown problem. In the learning process for aggregation of basic matchers' result, various pairs of compared ontologies must be used to successfully learn the neural network. For example, if the comments of entities are missed within the compared ontologies that were used for learning neural network, the results of basic matchers, which determine correspondences according the entities' comments, will have a minimal significance on the aggregation result for every matching process in the future although some pairs of compared ontologies contain entities' comments.

The ODGOMS [31] system does not have any aggregation method implemented. The system selects only one correspondence (one result) between two entities of different ontologies among nine results between these entities obtained by nine basic matchers. The basic matchers are arranged according to their importance that is determined by the authors. If the correspondence between two entities from the set of results obtained by the most important matcher satisfies the final alignment criteria, it is selected for the final alignment. If the correspondence does not satisfy the criteria, the process checks the correspondence result between these two entities by the second important matcher etc. Once a satisfying correspondence has been found, correspondences values between these two entities obtained by the remaining matchers are excluded from further consideration. It is possible that false correspondences between two entities enter final alignment if values calculated by the matchers considered as the most important strongly differ from the real values. Since overlapping of entities forming the correspondences is not allowed, high-quality correspondences (i.e., those corresponding to reality) obtained by basic matchers that are not highly ranked in the ODGOMS system will not be included into the final alignment.

Prior+ [32] is ontology mapping approach that uses parallel composition of basic matchers. The results are aggregated using the weighted aggregation method. Unlike the earlier approaches, the authors proposed the Harmony method that automatically determines the weighting factor of every basic matcher based on the results achieved by each matcher. The weighting factor of each matcher is determined according to the highest correspondences between two entities of compared ontologies.

We adopted the highest correspondences rule from [32] and proposed the Autoweight method [6] for automatically determining the weighting factors. In [6], the evaluation confirmed that Autoweight method achieved the best aggregation results. However, this method has a deficiency when determining which of the highest correspondences is trustworthy. For example, if the value of particular highest correspondence is 0.1, it can be assumed that this highest correspondence is not trustworthy because the similarity between two entities is only 0.1 and none of the values of remaining correspondences, in which one of the compared entities is involved, is not higher than this value. Therefore, we improved this method and proposed Autoweight++ method in [1]. This method does not consider the highest correspondences with lower values into calculation of the aggregation results. Furthermore, not only the determination of weighting factors is improved but also the weighted aggregation process. The problem of nonexistent correspondences (occur when a particular basic matcher is inapplicable, i.e., unable to produce results) decreases the final aggregated values of correspondences in the weighted aggregation. A nonexistent correspondence is replaced with the average of the existent correspondences between the two entities obtained by other basic matchers. Although the CroMatcher matching system described in [1,12,13] that uses the Autoweight++ aggregation achieved excellent results in the OAEI 2015 evaluation of the *Benchmark biblio ontology track*, which is the starting point for comparison of different matching systems, a new aggregation method for automatically determining the weighting factors, which is also based on the highest correspondences, is introduced in the newest version [11] of matching system described in [1]. Using this new aggregation method (Autoweight+++), matching system described in [1] achieved the best results in the OAEI 2016 evaluation of the Benchmark test set. However, we consider that there is still room for improvement of aggregation method therefore we implemented our new DWGA method for calculating weighting factors of basic matchers in the weighted aggregation. This method is presented in this paper and evaluated according to the *Benchmark biblio ontology track*. Furthermore, the detailed comparison of the DWGA method and the Autoweight+++ method introduced in [11], is also presented in this paper. As stated before, it is assumed that the comparison with Autoweight+++ method is a good indicator of real quality of new DGWA method.

There are several papers where the authors use genetic algorithm within the ontology matching process. The overview of employing the genetic algorithm (and other evolutionary algorithms) in ontology matching process is presented in [33].

An interesting solution, but not completely automatic, is presented in [34]. First, the user of the ontology matching system manually determines a small number of correspondences between compared ontologies. Considering these manually determined correspondences, the system adjusts the weighted factors of basic matchers and the complete matching proves is executed. As stated before, the approach is not completely automatic, and genetic algorithm adjusts weighted factors of basic matchers according to this small number of manually determined correspondences.

In [35], the authors propose the genetic algorithm approach that adjust optimal values of all parameters (thresholds, weighted factors etc.) within the matching system. A solution (chromosome) of genetic algorithm is the best combination of parameters' values that achieved the best matching results of an ontology matching system. The matching process is executed multiple times with different parameters' values. A fitness function evaluates a set of parameters' values comparing the obtained matching results with the known matching results of two ontologies. Hence, the parameters of a matching system are optimally adjusted only for pairs of compared ontologies for which the result is already known. As each ontology is unique, it is impossible to claim that the obtained parameters' values are the best not only for determining the correspondences between these tested pairs of ontologies, but also for determining correspondences between two ontologies for which the matching result does not exist. Our genetic algorithm approach determines the weighted factors of each basic matcher in the weighted aggregation of basic matcher based on results obtained by each basic matcher between currently compared ontologies.

Another approach in which the system optimally adjusts the weighted factors of basic matchers based on already known results is presented in [36]. This approach has the same problem of ontology uniqueness as the approach described above [35].

In [37], the authors propose a genetic algorithm approach to adjust the parameters of the matching system considering the matching results of the compared ontologies. The fitness function evaluates the matching results according to the most known evaluation measures: recall and precision [4]. As these measures evaluate matching results considering the known matching result between the compared ontologies, and these known results are not available, the authors propose two measures (pseudo-precision and pseudo-recall) that will substitute these real measures. Here, the problem is the estimation of approximate number of correct correspondences that must be known to get the values of precision and recall. Considering the pseudo-precision measure (number of correctly found correspondences over all found correspondences), the authors resolve this problem assuming that one entity of the first ontology is related only with one entity of the second ontology, therefore the system is adjusted to determine a set of correspondences with high similarity value where an entity is contained only within one correspondence. They assumed that the precision of this set of correspondences will be the highest possible (approximately 1). Furthermore, the exact number of correct correspondences between compared ontologies cannot be approximately determined, therefore the impact of the recall measure (number of correctly found correspondences over all correct correspondences) is decreased during the calculation of the fitness function. As the values of recall and precision are equally important when evaluating the matching results of an ontology matching system, decreased recall impact will affect the final matching result.

4. Genetic Algorithm

Genetic algorithm is a nature-inspired metaheuristic based on the process of natural selection [8]. As metaheuristic is a higher-level procedure that may quickly give a sufficiently satisfactory solution to an optimization problem, it can be used to resolve the optimization problem of the weighted factors assignment to the basic matchers in the weighted aggregation process. In this paper, a weighted aggregation method that uses 'elitist' genetic algorithm, which is one type of the genetic-inspired metaheuristic, to resolve the problem of determining the weighted factors is proposed.

The pseudo code of the 'elitist' genetic algorithm can be seen in Figure 3. The genetic algorithm starts with randomly generating the initial population of individuals (chromosomes). Each individual is called a chromosome and represents a potential solution of the optimization problem. Each chromosome consists of the set of genes that carry the information about the solution the chromosome represents. To evaluate how good each chromosome (i.e., its genes) represents the solution of the optimization problem, it is necessary to define the fitness function. The greatest challenge, when resolving some optimization problem by using the genetic algorithm, is the implementation of the fitness function. The fitness function represents the optimization problem that will be solved by using the genetic algorithm. If the fitness function is not defined well, then the whole evolution process of potential solutions within the genetic algorithm will not lead to a satisfactory solution of the actual optimization problem. The process of evolution is performed through several generations of potential solutions. While the condition of ending the evolution process is not satisfied, the more valuable chromosomes of actual population are selected during the selection process and the genetic operators' crossover and mutation are performed on these selected chromosomes to get better chromosomes than the selected ones. The selection process is important because it ensures that the chromosomes, for which the values of fitness function are high, remain in the further evolution process. The crossover genetic operator crosses the chromosomes with good genes to get a chromosome with better genes that the crossed ones. The mutation genetic operator usually mutates (changes) the value of one gene within the mutated chromosome, because of the possibility of creating a chromosome that has a higher value for fitness function than the mutated chromosome.

```
{
   Generate initial population of individuals (chromosomes) P;
   Evaluate the individuals from P;
   (While the condition of ending the evolution process is not satisfied)
      {
       Perform a selection of individuals from P;
       Perform a crossover operator over the selected individuals;
       Perform a mutation operator over the selected individuals;
       Create a new population and evaluate the individuals;
      }
   Return the best individual;
}
```

Figure 3. The pseudo code of genetic algorithm.

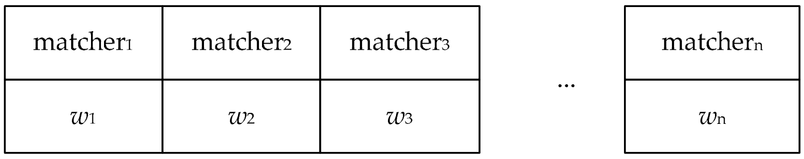
As stated before, the evolution process is repeated while the condition of ending process is not satisfied. The evolution process usually ends when the maximal number of generations is reached, but the process can also end earlier if the chromosome with a satisfied value of fitness function (i.e., the satisfied solution of optimization problem) is discovered. The chromosome with the highest value of the fitness function is selected as the solution of the current optimization problem.

4.1. Implementation of the Weighted Aggregation by Using Genetic Algorithm

In this section, the weighted aggregation that uses the genetic algorithm for automatically determining the weighted factors of basic matchers, is presented. The process of determining the weighted factors of basic matchers is described in detail as well as the usage of these weighted factors to get the aggregated values of correspondences between entities of matched ontologies.

4.1.1. The Chromosome

Before the start of the genetic algorithm, the chromosome that represents the solution of the optimization problem must be defined. Each chromosome consists of a certain number of genes, each of them containing a numeric value. All these numerical values form a solution to the problem. Therefore, considering the problem of determining optimal weighted factors of basic matchers in the weighted aggregation of basic matchers, each gene represents one weighted factor of the particular basic matcher. The example of the chromosome that contains genes which represents the weighted factors of basic matchers can be seen in Figure 4.



matcher_1	matcher_2	matcher_3		matcher_n
w_1	w_2	w_3	...	w_n

Figure 4. The chromosome for determining the weighted factors of string basic matchers.

Therefore, for each basic matcher there is a value (w_n) within chromosome that represents the weighted factor of the basic matcher. Each gene (w_n) can have values between 0 and 1. The sum of weighted factors in the weighted aggregation must be 1, therefore the attention should be paid on this condition during the random generating of genes' values. For example, if the first gene has the value 0.85, the sum of values of all other genes must have the value 0.15. The example of the correctly defined chromosome that consists of ten genes can contain the following values: $\text{gene}_1 = 0.1$, $\text{gene}_2 = 0.05$, $\text{gene}_3 = 0.05$, $\text{gene}_4 = 0.15$, $\text{gene}_5 = 0.25$, $\text{gene}_6 = 0.1$, $\text{gene}_7 = 0.05$, $\text{gene}_8 = 0.05$, $\text{gene}_9 = 0.1$, $\text{gene}_{10} = 0.1$. Therefore, when creating the initial population of chromosomes with random genes' values within the genetic algorithm process, this condition of the sum of weighted factors should be considered to get valid chromosomes.

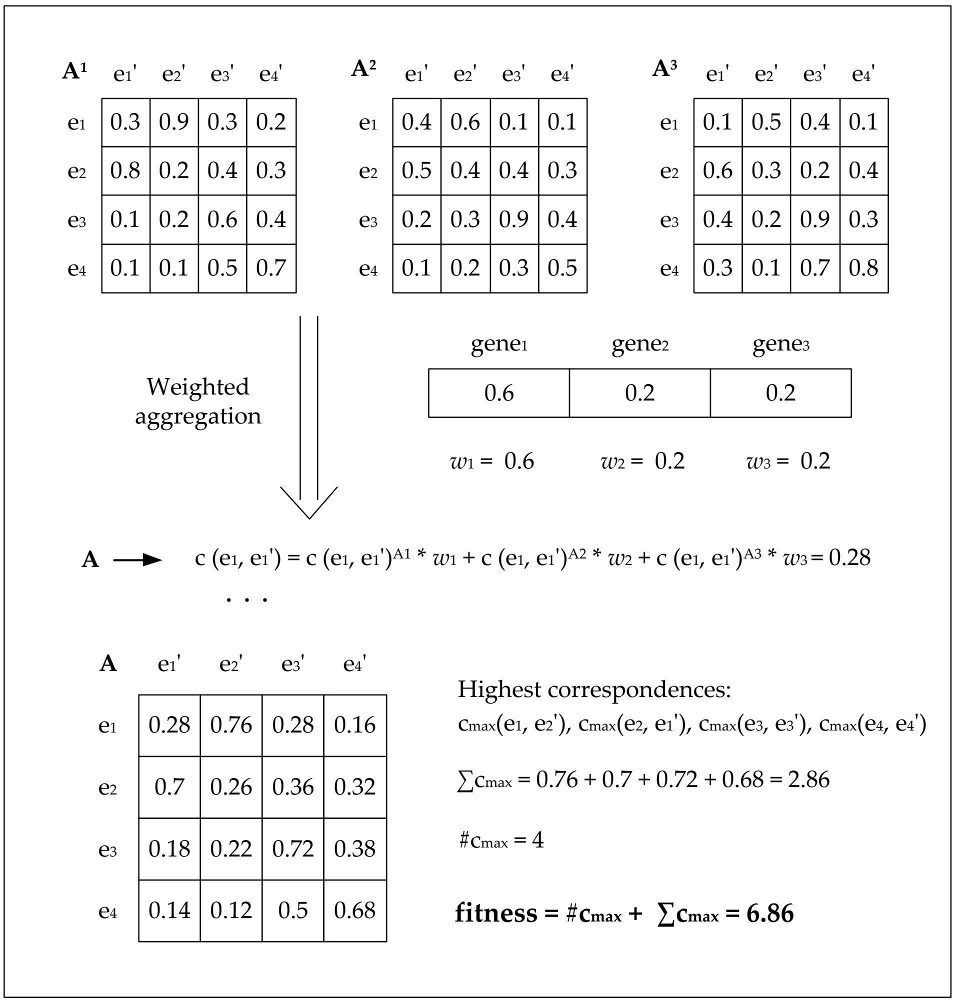
4.1.2. The Fitness Function

As stated before, the fitness function is the most important part of the genetic algorithm because it represents the optimization problem that must be solved. The fitness function evaluates the quality of each chromosome (solution) within the genetic algorithm process. Before the evaluation of a chromosome, the weighted aggregation of correspondences' results obtained by basic matchers is performed according to the weighted factors defined in this chromosome. The fitness function evaluates these aggregated results of correspondences to define the quality of chromosome's weighted factors. The fitness function is defined with the following equation:

$$\textit{fitness} = \#c_{\max} + \sum c_{\max} \tag{5}$$

where $\#c_{\max}$ is the number of the highest correspondences $c_{\max}$ (according to Definition 4) found within the aggregated matching results and the $\sum c_{\max}$ is the sum of values of these highest correspondences.

An example of the fitness function calculation for one chromosome can be seen in Figure 5. The chromosome consists of three genes which represent the weighted factors of three basic matchers in the weighted aggregation.



A¹	e1'	e2'	e3'	e4'
e1	0.3	0.9	0.3	0.2
e2	0.8	0.2	0.4	0.3
e3	0.1	0.2	0.6	0.4
e4	0.1	0.1	0.5	0.7

A²	e1'	e2'	e3'	e4'
e1	0.4	0.6	0.1	0.1
e2	0.5	0.4	0.4	0.3
e3	0.2	0.3	0.9	0.4
e4	0.1	0.2	0.3	0.5

A³	e1'	e2'	e3'	e4'
e1	0.1	0.5	0.4	0.1
e2	0.6	0.3	0.2	0.4
e3	0.4	0.2	0.9	0.3
e4	0.3	0.1	0.7	0.8

gene1	gene2	gene3
0.6	0.2	0.2

$w_1 = 0.6$ $w_2 = 0.2$ $w_3 = 0.2$

$\mathbf{A} \longrightarrow$ $c(e_1, e_1') = c(e_1, e_1')^{A1} * w_1 + c(e_1, e_1')^{A2} * w_2 + c(e_1, e_1')^{A3} * w_3 = 0.28$

. . .

A	e1'	e2'	e3'	e4'
e1	0.28	0.76	0.28	0.16
e2	0.7	0.26	0.36	0.32
e3	0.18	0.22	0.72	0.38
e4	0.14	0.12	0.5	0.68

Highest correspondences:
$c_{\max}(e_1, e_2')$, $c_{\max}(e_2, e_1')$, $c_{\max}(e_3, e_3')$, $c_{\max}(e_4, e_4')$

$\sum c_{\max} = 0.76 + 0.7 + 0.72 + 0.68 = 2.86$

$\#c_{\max} = 4$

fitness = $\#c_{\max} + \sum c_{\max}$ = 6.86

Figure 5. The calculation of the fitness function.

The values of genes are 0.6, 0.2 and 0.2. The matching results of these three basic matchers are presented in matrices A^1, A^2 and A^3 which contain the values of all correspondences between any two entities of compared ontologies obtained by these three matchers. The entities of the first ontology are e_1, e_2, e_3 and e_4, while the entities of the second ontology are e_1', e_2', e_3' and e_4'. The matching results of basic matchers are aggregated with weighted aggregation of basic matchers using the weighted factors of the chromosome. After performing the weighted aggregation, the matrix A contains the aggregated values of all correspondences between any two entities of the compared ontologies. Considering the aggregated results, the value of the fitness function must be calculated. In this example, there are four highest correspondences in the aggregated results: $c_{\max}$ (e_1, e_2'), $c_{\max}$ (e_2, e_1'), $c_{\max}$ (e_3, e_3') and $c_{\max}$ (e_4, e_4'). The sum of values of these four correspondences is 2.86. Hence, the value of fitness function is 6.86. The higher the value of the fitness function, the more suitable the genes (weighted factors) of proposed chromosome are for executing the weighted aggregation of basic matchers. Therefore, those weighted factors (genes) of a chromosome that have the highest value of the fitness function in the genetic algorithm process, are selected as the solution for executing the weighted aggregation of basic matchers in the ontology matching process.

4.1.3. Selection

As stated before, the selection is the process of choosing chromosomes from the previous generation on which the crossover and mutation genetic operators are going to be performed to obtain even better chromosomes than those of the previous generation. In this way, the natural selection where better individuals survive, is simulated. Although most selected chromosomes have a high value of the fitness function, it is a good practice to allow, through the selection process, that sometimes a chromosome with low fitness function value enters the next generation. Although it might not have a high value of fitness function, it can contain some good gene value that can help to create a better final chromosome, which represents a solution of the optimization problem. There are different types of selection [38], but in this optimization problem of determining the optimal weighted factors for weighted aggregation of basic matchers, the deterministic tournament selection is chosen. The deterministic tournament selection randomly selects k chromosomes from the pool of chromosomes. The chromosome with the highest fitness function value is selected in the next generation. This process is repeated n times, as the next generation consists of n chromosomes. For example, let the generation consists of 20 chromosomes. After the execution of genetic operators of crossover and mutation, there are many new chromosomes (pool of chromosomes) among which the deterministic tournament selection is performed. Let each chromosome consists of 5 genes which values represent weighted factors of 5 basic matcher. The fitness function of each chromosome is calculated according to Equation (5) defined in Section 4.1.2. As each generation consists of 20 chromosomes, the process of deterministic tournament selection is repeated 20 times to get 20 chromosomes for the new generation. Let the number k of randomly selected chromosomes in each tournament selection is set to 5. Thus, in each selection process, 5 randomly selected chromosomes enter the tournament selection process and only the chromosome with the highest fitness value enters the next generation.

It may happen that this tournament selection does not include in the next generation the chromosome with the highest fitness function value of the previous generation due to random selection of k chromosomes within n iterations of tournament selection. Therefore, in this optimization problem the elitist selection mechanism is activated. This mechanism always selects the chromosome with the highest fitness function value of the previous generation to be a part of the next generation regardless of chosen type of selection in the evolution process and thus this elitist selection mechanism guarantees that the solution quality obtained by the genetic algorithm will not decrease from one generation to the next.

4.1.4. Genetic Operators

The genetic operators provide the creation of even better chromosomes (solutions) through the evolution process of the genetic algorithm. The implementation of crossover and mutation genetic operators, used for optimization problem of determining the optimal weighted factors for weighted aggregation of basic matchers, is described below. The crossover operator crosses the genes of two selected chromosomes to get new chromosomes that will have better fitness function value than its parents, i.e., crossed chromosomes. The mutation operator usually changes the value of one gene within selected chromosome to create a new chromosome.

Crossover. The crossover is a process of creating two new chromosomes based on the crossing of two chromosomes. As stated before, the crossover operator is performed to get new chromosomes that will have better fitness function value than the crossed chromosomes. Before doing the crossover, a crossover point must be defined. The crossover point is the position within chromosomes from which the chromosomes do the crossing, i.e., interchange the values of genes. Before the crossover point, the chromosomes remain unchanged. This way, two new chromosomes are created, one with the genes' values of the first crossed chromosome before the crossover point and the genes' values of the second crossed chromosome after the crossover point, and another with the genes' values of the second crossed chromosome before the crossover point and the genes' values of the first crossed chromosome after the crossover point. There are different types of crossover with more than one crossover point (usually two), but in this paper the crossover genetic operator with one crossover point is implemented, as the results of the genetic algorithm process were equal considering the usage of one or more crossover points. Due to equal results of different types of crossover, the crossover with one crossover point is used, because it performs faster than any other type of crossover. The example of crossover of two chromosomes with one crossover point between fifth and sixth gene is presented in Figure 6.

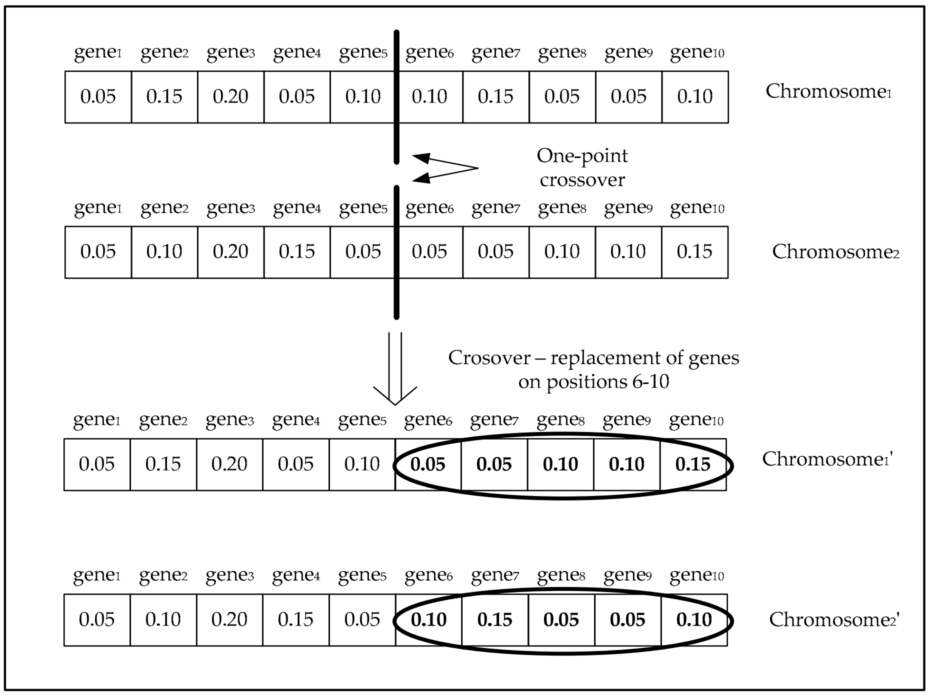

gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.10 0.15 0.05 0.05 0.10
Chromosome1
One-point crossover
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.10 0.20 0.15 0.05 0.05 0.05 0.10 0.10 0.15
Chromosome2
Crosover – replacement of genes on positions 6-10
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.05 0.05 0.10 0.10 0.15
Chromosome1'
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.10 0.20 0.15 0.05 0.10 0.15 0.05 0.05 0.10
Chromosome2'

Figure 6. The example of crossover with one crossover point.

At the beginning, there are two chromosomes with the following genes' values: chromosome$_1$ (0.05, 0.15, 0.2, 0.05, 0.10, 0.10, 0.15, 0.05, 0.05, 0.10), chromosome$_2$ (0.05, 0.10, 0.20, 0.15, 0.05, 0.05, 0.05, 0.10, 0.10, 0.15). The random-generated crossover point is found between fifth and sixth gene. Therefore, when crossing two chromosomes, the genes from the sixth to the tenth position are interchanged and thus two new chromosomes are created: chromosome$_1$′ (0.05, 0.15, 0.2, 0.05, 0.10, 0.05, 0.05, 0.10, 0.10, 0.15), and chromosome$_2$′ (0.05, 0.10, 0.20, 0.15, 0.05, 0.10, 0.15, 0.05, 0.05, 0.10).

The problem of crossing two chromosomes whose genes represent weighted factors of basic matchers is that after the crossing process is done, there is a possibility that the sum of gene values in one chromosome is greater than 1, and in another one it is less than 1. As stated before, the sum of weighted factors in the weighted aggregation must be 1. An example of resolving this problem can be seen in Figure 7.

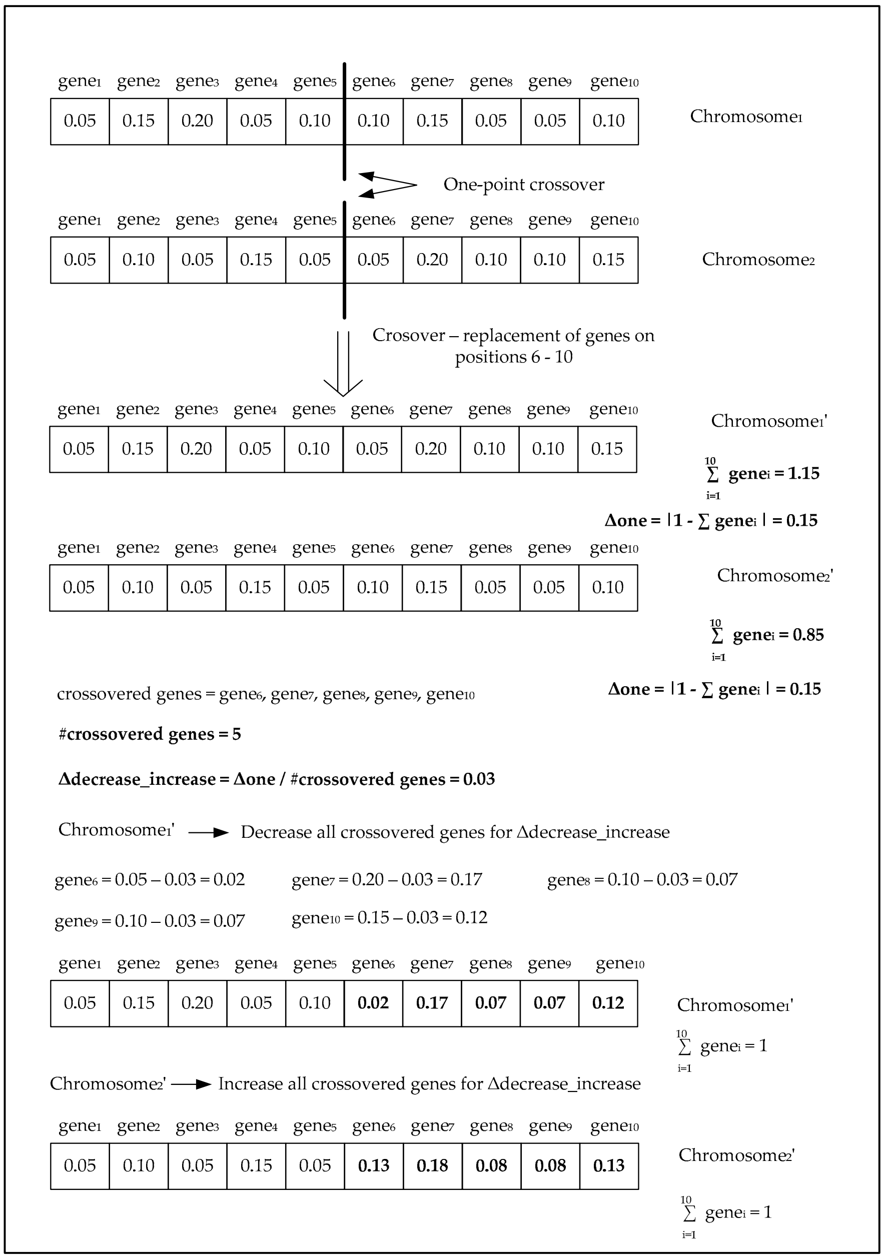

gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.10 0.15 0.05 0.05 0.10
Chromosome1
One-point crossover
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.10 0.05 0.15 0.05 0.05 0.20 0.10 0.10 0.15
Chromosome2
Crosover – replacement of genes on positions 6 - 10
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.05 0.20 0.10 0.10 0.15
Chromosome1'
$\sum_{i=1}^{10}$ genei = 1.15
Δone = |1 - ∑ genei | = 0.15
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.10 0.05 0.15 0.05 0.10 0.15 0.05 0.05 0.10
Chromosome2'
$\sum_{i=1}^{10}$ genei = 0.85
Δone = |1 - ∑ genei | = 0.15
crossovered genes = gene6, gene7, gene8, gene9, gene10
#crossovered genes = 5
Δdecrease_increase = Δone / #crossovered genes = 0.03
Chromosome1' Decrease all crossovered genes for Δdecrease_increase
gene6 = 0.05 – 0.03 = 0.02
gene7 = 0.20 – 0.03 = 0.17
gene8 = 0.10 – 0.03 = 0.07
gene9 = 0.10 – 0.03 = 0.07
gene10 = 0.15 – 0.03 = 0.12
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.02 0.17 0.07 0.07 0.12
Chromosome1'
$\sum_{i=1}^{10}$ genei = 1
Chromosome2' Increase all crossovered genes for Δdecrease_increase
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.10 0.05 0.15 0.05 0.13 0.18 0.08 0.08 0.13
Chromosome2'
$\sum_{i=1}^{10}$ genei = 1

Figure 7. One-point crossover with increased/decreased values of crossovered genes.

At the beginning, there are two chromosomes: chromosome$_1$ (0.05, 0.15, 0.2, 0.05, 0.10, 0.10, 0.15, 0.05, 0.05, 0.10), and chromosome$_2$ (0.05, 0.10, 0.05, 0.15, 0.05, 0.05, 0.20, 0.10, 0.10, 0.15). The random-generated crossover point is found between the fifth and the sixth gene. After the crossover process, two new chromosomes are created: chromosome$_1'$ (0.05, 0.15, 0.2, 0.05, 0.10, 0.05, 0.20, 0.10, 0.10, 0.15), and chromosome$_2'$ (0.05, 0.10, 0.05, 0.15, 0.05, 0.10, 0.15, 0.05, 0.05, 0.10). The sum of genes' values $\sum$gene$_i$ of the first chromosome (chromosome$_1'$) is equal to 1.15 while the sum of genes' values of the second chromosome (chromosome$_2'$) is equal to 0.85. Hence, the absolute difference Δone between the requested sum of weighted factors' values (which must be equal to 1) and $\sum$ gene$_i$ of any of the two newly created chromosomes is equal to 0.15. In the first chromosome, the sum of genes is equal to 1.15, therefore the values of the crossed genes (which are found from the sixth to the tenth position within the chromosome) must be decreased by a total value of 0.15. Considering that the values of five genes must be decreased, all values of these five genes must be decreased for 0.03 which is the value of parameter Δdecrease_increase. After decreasing the values of crossed genes, the genes' values of chromosome$_1$' are (0.05, 0.15, 0.2, 0.05, 0.10, 0.02, 0.17, 0.07, 0.07, 0.12). The same process is done for the chromosome$_2$' but in this case the values of crossed genes are increased for 0.03 (Δdecrease_increase), therefore the genes' values of chromosome$_2$' are (0.05, 0.10, 0.05, 0.15, 0.05, 0.13, 0.18, 0.08, 0.08, 0.13). The process of crossover genetic operator can be defined in three steps as follows:

- Randomly generate the crossover point.
- Make a crossover of two chromosomes interchanging the values of genes after the crossover point.
- Check the sum of genes' values $\sum gene_i$ within the newly created chromosomes. If the sum of genes' values is not equal to 1 then:
    - ○ Calculate the absolute difference Δone between 1 and the sum of genes' values $\sum gene_i$
    - ○ Calculate the value Δdecrease_increase, which is equal to quotient of the absolute difference Δone and number of genes after the crossover point
    - ○ Decrease (or increase) all genes' values that are found after the defined crossover point for Δdecrease_increase value to adjust the sum of genes' values $\sum gene_i$ to be equal to 1.

Mutation. The mutation genetic operator changes (mutates) the value of one or more genes within the chromosomes. The change of only one gene value within the chromosome can completely change the characteristics of this chromosome (i.e., the value of the fitness function) and improve the evolution process of searching for the optimal solution. Before the mutation process, the gene whose value will be changed must be selected randomly. Considering the problem of the sum of genes' values $\sum gene_i$ that must be equal to 1, in this paper the mutation operator is modified to adapt to the optimization problem that must be solved. Thus, the mutation is performed on two genes within the chromosome in a way that the values of these two genes are interchanged. As stated before, the genes whose values will be interchanged are randomly selected. The interchange of two genes' values preserves the correctness of the chromosome (the sum of genes' values $\sum gene_i$ stays equal to 1), but also changes the chromosome characteristics and the value of the fitness function. The example of the implemented mutation operator can be seen in Figure 8.

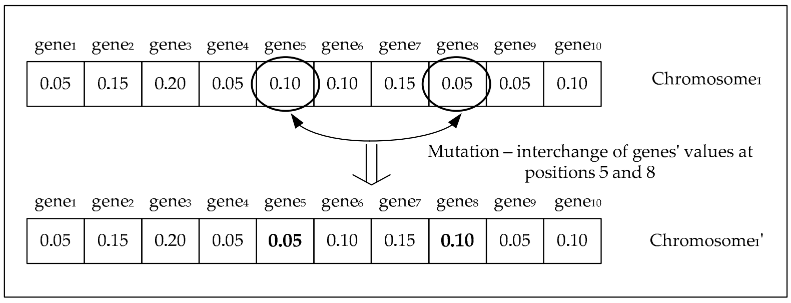

gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.10 0.10 0.15 0.05 0.05 0.10
Chromosome1
Mutation – interchange of genes' values at positions 5 and 8
gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
0.05 0.15 0.20 0.05 0.05 0.10 0.15 0.10 0.05 0.10
Chromosome1'

Figure 8. The example of the mutation genetic operator.

At the beginning there is one chromosome: chromosome$_1$ (0.05, 0.15, 0.2, 0.05, 0.10, 0.10, 0.15, 0.05, 0.05, 0.10). Then, two genes are randomly selected to make the interchange mutation process. In this example, the genes at positions 5 and 8 are selected for mutation. After the mutation process, a new chromosome is created: chromosome$_1$' (0.05, 0.15, 0.2, 0.05, 0.05, 0.10, 0.15, 0.10, 0.05, 0.10). The mutation process can be defined in two steps as follows:

- Randomly select two genes within the chromosome on which the mutation will be performed.
- Interchange the values of the selected genes within the chromosome.

4.2. Evolution Process in the Genetic Algorithm

As stated before, the process of evolution is performed through several generations of potential solutions. While the condition of ending the evolution process is not satisfied, the selection, crossover and mutation are performed on every generation to obtain a satisfactory solution of the requested optimization problem. In the implementation of the genetic algorithm for determining the optimal weighted factors of basic matchers, two ending conditions are set. The first ending condition stops the evolution process when the maximal number of generations is reached, while the second ending condition stops the process when there is no better solution found in max_{same} consecutive generations of potential solutions. This second ending solution speeds up the performance of the genetic algorithm when a satisfied solution of optimization problem is discovered. The complete pseudo code of the genetic algorithm for determining the optimal weighted factors can be seen in Figure 9. In the next section, the evaluation of the genetic algorithm for determining the optimal weighted factors of basic matchers implemented as a component of the CroMatcher system is presented.

```
{
  Randomly generate the initial population of chromosomes (individuals) P,
  making sure that the sum of all genes' values within each generated
  chromosome is equal to 1;

  Evaluate the individuals from P (fitness function defined in section
  4.1.2.)

  (While at least one of two defined conditions of ending the evolution
   process <$max_{same}$ or maximal number of evolution> is not satisfied)

  {
        Perform a selection of chromosomes from P (deterministic tournament
        selection defined in section 4.1.3.)

        Perform a crossover operator over the selected chromosomes
        (crossover operator defined in section 4.1.4.)

        Perform a mutation operator over the selected chromosomes (mutation
        operator defined in section 4.1.4.)

        Create a new population and evaluate new individuals (fitness
        function defined in section 4.1.2.)
  }

  Return the best chromosome (containing the best found values of weighted
  factors according to the defined fitness value)
}
```

Figure 9. The pseudo code of genetic algorithm for determining the weighted factors of basic matchers.

5. Evaluation

The weighted aggregation with DGWA method, as a part of CroMatcher system, was implemented in Java (JDK 1.7) using Netbeans IDE 7.4. The java library Java Genetic Algorithm and Programming (JGAP) is used for genetic algorithm implementation within the proposed aggregation method. The experimental testing of the proposed method was performed with the constant size of a population which totaling 1000 chromosomes and the constant number of evolutions set to 150 evolutions with a probability of crossover of 0.70 and a probability of mutation of 0.05. The deterministic tournament selection with the elitist selection mechanism is used. There are 2 ending solutions of the genetic algorithm process: (1) when the maximal number of generations (150) is reached, and (2) when there is no better solution found in 10 consecutive generations of potential solutions. This configuration of the genetic algorithm has shown the best results and any enlargement of the population size as well as the number of evolutions has not improved the evaluation results. The evaluation was performed on *Benchmark biblio ontology track* [10]. The *Benchmark biblio ontology track* is the largest test set in the evaluation of the ontology matching systems organized by OAEI (Ontology Alignment Evaluation Initiative) [14,15]. This test case contains more than 100 pairs of ontologies (written in OWL) and the alignment results between them. In each pair of ontologies, the first ontology has all the information related to a specific domain. In the second ontology, certain components (labels and comments of entities, ontology structure, properties, etc.) are not defined in order to test matching systems on different test cases. Thus, the advantages and disadvantages of each matching system can be examined according to the missing components in a particular test. A new weighted aggregation method that uses the DWGA method will be evaluated as a part of the CroMatcher ontology matching system [1] because this system produced the best matching results for *Benchmark biblio test case* in OAEI 2016 [39], thus is relevant for the evaluation. The CroMatcher system contains three weighted aggregations: the aggregation of string basic matchers, the aggregation of structure basic matchers and the aggregation of aggregated results obtained by the first two aggregations. The weighted aggregation is a particularly important part of this system and the aggregation must be implemented well to produce quality matching results at the end of matching process. Thus, the evaluation of the DGWA method within the CroMatcher system will show the efficiency of this method as part of the matching system. Furthermore, the comparison between the results produced by CroMatcher using the weighted aggregation with DWGA method and CroMatcher using the weighted aggregation with Autoweight+++ method (OAEI 2016) will be presented in detail. We compare these two methods for determining weighted factors because the weighted aggregation, even with the first version of Autoweight+++ method (Autoweight method), shows the best results of all aggregation methods tested in [6] and also this weighted aggregation with Autoweight+++ method is a part of CroMatcher that produces the best results for *Benchmark biblio test case* in OAEI 2016 as stated before. The evaluation measures that we use to compare the results produced by two versions of CroMatcher system (one with DGWA method and another with Autoweight+++ method) are the following:

- **Precision**, which is the ratio of correctly found correspondences over the total number of correspondences returned by the matching system,
- **Recall**, which is the ratio of correctly found correspondences over the total number of all correct correspondences between two ontologies,
- **F-Measure**, which is the harmonic mean of precision and recall.

F-Measure is the most important measure for the evaluation of the matching process because it combines the values of precision and recall. There is a larger number of correct correspondences when the recall value is high. Considering the precision measure, there are less false correspondences when its value is high. Therefore, if the value of F-Measure is high, there is less additional work for the expert to correct obtained correspondences (finding additional correct correspondences and deleting the false found correspondences). Considering the *Benchmark biblio ontology track*, there are special test cases that consist of ontologies in which some of the main components (labels and comments of entities, ontology structure, properties, etc.) are partially implemented or are not implemented at all. Consequently, the recall value will be low (small number of correctly found correspondences) but then it is important that the number of false correspondences that system finds is not large to preserve a high value of precision. These test cases with partially defined components within compared ontologies evaluate the matching system according to the precision value. If the precision value is high in this situation (i.e., the system does not determine many false correspondences), the system determines well which correspondence is correct or false. As stated before, the goal of the matching system is to achieve both recall and precision values as high as possible to have less additional work of correcting obtained correspondences after the matching process. Furthermore, the values of weighted factors obtained by the DGWA and Autoweight+++ method for each test case are compared. At the end of evaluation, the values of correspondences between entities of matched ontologies produced by two versions of CroMatcher system are compared too to determine the reliability of these correspondences. The higher the values of found correspondences between matched ontologies, the more reliable these correspondences are i.e., there is a higher probability that these correspondences are not false. A version of CroMatcher with the new DGWA method produces great alignment results with very high confidence values of found correspondences and it is the most important progress in matching process in comparison to CroMatcher with Autoweight+++ method, which does not produce so reliable correspondences. A complete evaluation is described below.

In Table 1, the results of each test case within the *Benchmark biblio ontology track* produced by two versions of CroMatcher system (one with Autoweight+++ method and another one with the DGWA method) are presented. The overall results of these two versions of matching system are almost equal, but there are quite different results for many test cases produced by these two versions of system, proving our assumption that there is still room for improvement in matching *Benchmark biblio ontology track*. It also justifies the direction of this research for creating a new aggregation method.

Table 1. The results of each test case within the *Benchmark biblio ontology track* obtained by two versions of CroMatcher system (Autoweight+++ method vs. DGWA method).

	Autoweight+++			**DWGA**				**Autoweight+++**			**DWGA**		
Test	**R**	**P**	**F-M**	**R**	**P**	**F-M**	**Test**	**R**	**P**	**F-M**	**R**	**P**	**F-M**
101	1.000	1.000	1.000	1.000	1.000	1.000	258-4	0.792	**0.963**	**0.869**	0.792	0.950	0.864
201	1.000	1.000	1.000	1.000	1.000	1.000	258-6	0.719	**0.959**	0.822	**0.730**	0.946	**0.824**
201-2	1.000	1.000	1.000	1.000	1.000	1.000	258-8	0.563	**0.844**	0.675	**0.636**	0.836	**0.722**
201-4	1.000	1.000	1.000	1.000	1.000	1.000	259	0.568	**0.798**	0.664	**0.660**	0.753	**0.703**
201-6	1.000	1.000	1.000	1.000	1.000	1.000	259-2	**0.897**	**0.967**	**0.931**	0.887	0.946	0.916
201-8	1.000	1.000	1.000	1.000	1.000	1.000	259-4	0.846	0.954	0.897	**0.887**	**0.967**	**0.925**
202	**0.866**	0.944	**0.903**	0.856	0.944	0.898	259-6	**0.825**	**0.920**	**0.870**	0.712	0.832	0.767
202-2	0.980	1.000	0.990	0.980	1.000	0.990	254-6	0.728	1.000	0.843	0.728	1.000	0.843
202-4	0.918	0.989	0.952	**0.928**	**0.990**	**0.958**	254-8	**0.637**	**1.000**	**0.778**	0.485	0.942	0.640
202-6	**0.887**	**0.978**	**0.930**	0.866	0.955	0.908	257	0.334	**0.847**	**0.479**	0.334	0.459	0.387
202-8	0.877	0.989	0.930	0.877	0.989	0.930	257-2	**0.819**	1.000	**0.900**	0.788	1.000	0.881
221	1.000	1.000	1.000	1.000	1.000	1.000	257-4	**0.758**	1.000	**0.862**	0.728	1.000	0.843
222	1.000	1.000	1.000	1.000	1.000	1.000	257-6	**0.516**	**0.945**	**0.668**	0.425	0.934	0.584
223	1.000	1.000	1.000	1.000	1.000	1.000	257-8	**0.455**	**0.883**	**0.601**	0.273	0.819	0.410
224	1.000	1.000	1.000	1.000	1.000	1.000	258	0.563	**0.886**	0.688	**0.605**	0.841	**0.704**
225	1.000	1.000	1.000	1.000	1.000	1.000	258-2	0.886	0.978	0.930	**0.917**	0.978	**0.947**
228	1.000	1.000	1.000	1.000	1.000	1.000	258-4	0.792	**0.963**	**0.869**	0.792	0.950	0.864
232	1.000	1.000	1.000	1.000	1.000	1.000	258-6	0.719	**0.959**	0.822	**0.730**	0.946	**0.824**
233	1.000	1.000	1.000	1.000	1.000	1.000	258-8	0.563	**0.844**	0.675	**0.636**	0.836	**0.722**
236	1.000	1.000	1.000	1.000	1.000	1.000	259	0.568	**0.798**	0.664	**0.660**	0.753	**0.703**
237	1.000	1.000	1.000	1.000	1.000	1.000	259-2	**0.897**	**0.967**	**0.931**	0.887	0.946	0.916
238	1.000	1.000	1.000	1.000	1.000	1.000	259-4	0.846	0.954	0.897	**0.887**	**0.967**	**0.925**
239	1.000	1.000	1.000	1.000	1.000	1.000	259-6	**0.825**	**0.920**	**0.870**	0.712	0.832	0.767
240	1.000	1.000	1.000	1.000	1.000	1.000	254-6	0.728	1.000	0.843	0.728	1.000	0.843
241	1.000	1.000	1.000	1.000	1.000	1.000	254-8	**0.637**	**1.000**	**0.778**	0.485	0.942	0.640
246	1.000	1.000	1.000	1.000	1.000	1.000	257	0.334	**0.847**	**0.479**	0.334	0.459	0.387
247	1.000	1.000	1.000	1.000	1.000	1.000	257-2	**0.819**	1.000	**0.900**	0.788	1.000	0.881
248	0.846	0.932	0.887	0.846	0.932	0.887	257-4	**0.758**	1.000	**0.862**	0.728	1.000	0.843
248-2	0.980	1.000	0.990	**0.990**	1.000	**0.995**	257-6	**0.516**	**0.945**	**0.668**	0.425	0.934	0.584
248-4	0.928	1.000	0.963	0.928	1.000	0.963	257-8	**0.455**	**0.883**	**0.601**	0.273	0.819	0.410
248-6	0.897	1.000	0.946	0.897	1.000	0.946	258	0.563	**0.886**	0.688	**0.605**	0.841	**0.704**
248-8	0.877	0.989	0.930	0.877	0.989	0.930	258-2	0.886	0.978	0.930	**0.917**	0.978	**0.947**
249	0.743	0.889	0.809	**0.763**	**0.914**	**0.832**	258-4	0.792	**0.963**	**0.869**	0.792	0.950	0.864
249-2	0.990	1.000	0.995	0.990	1.000	0.995	258-6	0.719	**0.959**	0.822	**0.730**	0.946	**0.824**
249-4	0.959	1.000	0.979	0.959	1.000	0.979	258-8	0.563	**0.844**	0.675	**0.636**	0.836	**0.722**
249-6	0.825	0.931	0.875	0.825	0.931	0.875	259	0.568	**0.798**	0.664	**0.660**	0.753	**0.703**
249-8	0.774	0.863	0.816	**0.794**	**0.906**	**0.846**	259-2	**0.897**	**0.967**	**0.931**	0.887	0.946	0.916
250	**0.576**	0.950	**0.717**	0.546	**1.000**	0.706	259-4	0.846	0.954	0.897	**0.887**	**0.967**	**0.925**
250-2	0.910	1.000	0.953	0.910	1.000	0.953	259-6	**0.825**	**0.920**	**0.870**	0.712	0.832	0.767
250-4	0.819	1.000	0.900	**0.849**	1.000	**0.918**	254-6	0.728	1.000	0.843	0.728	1.000	0.843
250-6	0.697	1.000	0.821	**0.728**	1.000	**0.843**	254-8	**0.637**	**1.000**	**0.778**	0.485	0.942	0.640
250-8	0.607	1.000	0.755	**0.637**	1.000	**0.778**	257	0.334	**0.847**	**0.479**	0.334	0.459	0.387
251	0.709	0.945	0.810	**0.730**	**0.946**	**0.824**	257-2	**0.819**	1.000	**0.900**	0.788	1.000	0.881
251-2	0.907	1.000	0.951	**0.948**	1.000	**0.973**	257-4	**0.758**	1.000	**0.862**	0.728	1.000	0.843
251-4	0.803	0.975	0.881	**0.875**	**0.989**	**0.929**	257-6	**0.516**	**0.945**	**0.668**	0.425	0.934	0.584

Table 1. *Cont.*

	Autoweight+++			DWGA				Autoweight+++			DWGA		
Test	**R**	**P**	**F-M**	**R**	**P**	**F-M**	**Test**	**R**	**P**	**F-M**	**R**	**P**	**F-M**
251-6	0.792	0.963	0.869	**0.834**	**0.988**	**0.904**	257-8	**0.455**	**0.883**	**0.601**	0.273	0.819	0.410
251-8	**0.813**	**0.988**	**0.892**	0.782	0.962	0.863	258	0.563	**0.886**	0.688	**0.605**	0.841	**0.704**
252	0.743	**0.879**	0.805	**0.774**	0.853	**0.812**	258-2	0.886	0.978	0.930	**0.917**	0.978	**0.947**
252-2	0.949	1.000	0.974	0.949	1.000	0.974	258-4	0.792	**0.963**	**0.869**	0.792	0.950	0.864
252-4	**0.877**	0.966	**0.919**	0.866	0.966	0.913	258-6	0.719	**0.959**	0.822	**0.730**	0.946	**0.824**
252-6	**0.877**	**0.956**	**0.915**	0.846	0.943	0.892	258-8	0.563	**0.844**	0.675	**0.636**	0.836	**0.722**
252-8	**0.825**	0.953	**0.884**	0.794	**0.963**	0.870	259-8	**0.660**	**0.811**	**0.728**	0.609	0.787	0.687
253	0.691	0.828	0.753	**0.722**	**0.865**	**0.787**	260	**0.438**	0.824	**0.572**	0.407	**0.929**	0.566
253-2	0.939	1.000	0.969	**0.949**	1.000	**0.974**	260-2	0.875	1.000	0.933	0.875	1.000	0.933
253-4	0.897	1.000	0.946	**0.908**	1.000	**0.952**	260-4	0.813	1.000	0.897	0.813	1.000	0.897
253-6	0.877	**0.989**	**0.930**	0.877	0.978	0.925	260-6	**0.688**	1.000	**0.815**	0.657	1.000	0.793
253-8	**0.784**	**0.905**	**0.840**	0.743	0.858	0.796	260-8	**0.469**	1.000	**0.639**	0.438	1.000	0.609
254	**0.576**	0.950	**0.717**	0.546	**1.000**	0.706	261	0.364	0.601	0.453	0.364	**0.924**	**0.522**
254-2	0.879	1.000	0.936	0.879	1.000	0.936	261-2	0.819	0.900	0.858	0.819	**0.965**	**0.886**
254-4	0.819	1.000	0.900	0.819	1.000	0.900	261-4	0.667	0.880	0.759	**0.697**	**0.885**	**0.780**
254-6	0.728	1.000	0.843	0.728	1.000	0.843	261-6	0.667	0.786	0.722	**0.758**	**0.893**	**0.820**
254-8	**0.637**	**1.000**	**0.778**	0.485	0.942	0.640	261-8	0.546	0.721	0.621	0.546	**0.948**	**0.693**
257	0.334	**0.847**	**0.479**	0.334	0.459	0.387	262	0.273	**0.693**	0.392	**0.364**	0.500	**0.421**
257-2	**0.819**	1.000	**0.900**	0.788	1.000	0.881	262-2	0.879	1.000	0.936	0.879	1.000	0.936
257-4	**0.758**	1.000	**0.862**	0.728	1.000	0.843	262-4	**0.667**	1.000	**0.800**	0.637	1.000	0.778
257-6	**0.516**	**0.945**	**0.668**	0.425	0.934	0.584	262-6	0.607	1.000	0.755	0.607	1.000	0.755
257-8	**0.455**	**0.883**	**0.601**	0.273	0.819	0.410	262-8	**0.485**	1.000	**0.653**	0.455	1.000	0.625
258	0.563	**0.886**	0.688	**0.605**	0.841	**0.704**	265	0.000	**1.000**	0.000	**0.094**	0.301	**0.143**
258-2	0.886	0.978	0.930	**0.917**	0.978	**0.947**	266	0.152	0.358	0.213	**0.243**	**0.471**	**0.321**

To prove the quality of CroMatcher system version with DGWA method in different matching conditions, similar test cases within the *Benchmark biblio ontology track* are grouped according to the implemented components within ontologies of these test sets to evaluate the version of the CroMatcher with DGWA method in different environments. There are 5 main different groups of test cases that consist of ontologies in which: all main components are implemented (ALL), the labels and comments of entities are missing (No-LC), the labels, comments and ontology structure are missing (No-LCS), the labels, comments and instances of entities are missing (No-LCI) and the labels, comments and properties are missing (No-LCP). A comparison between two versions of CroMatcher system (one with Autoweight+++ method and another with the DGWA method) is made according to these 5 different groups of test cases.

Considering the recall measure, the comparison of the results obtained by CroMatcher with the Autoweight+++ method and CroMatcher with the DGWA method for different groups of test cases within *Benchmark biblio ontology track* can be seen in Figure 10.

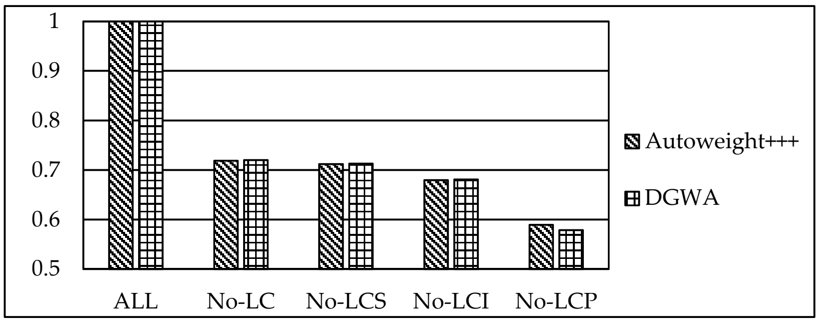


Figure 10. Comparison of methods Autoweight+++ and DGWA according to the results of recall measure for different groups of test cases within *Benchmark biblio ontology track.*

Here, the CroMatcher with the DGWA method achieved better results of recall than the CroMatcher with the Autoweight+++ method for three groups of test cases (No-LC, No-LCS and No-LCI). The CroMatcher with the Autoweight+++ method achieved better results of recall for No-LCP group of test cases while the results are equal for ALL group of test cases. CroMatcher with the DGWA method found more correct correspondences (higher recall value) within the final matching results than CroMatcher with the Autoweight+++ method. However, the results are almost equal for all groups of test cases, therefore it can be assumed that both versions of system produce satisfactory results for the entire *Benchmark biblio ontology track*. Both versions of system achieve almost equal results for precision too. The result of precision can be seen in Figure 11.

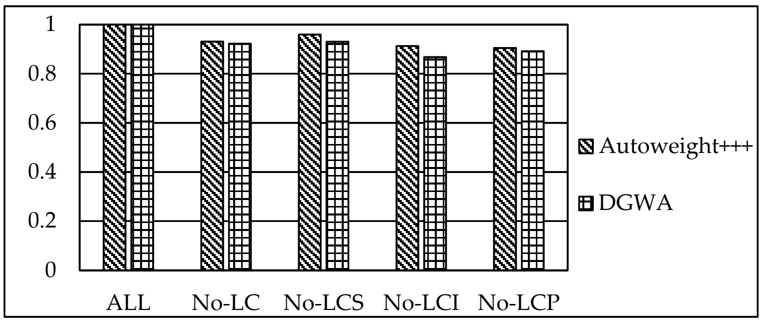


Figure 11. Comparison of methods Autoweight+++ and DGWA according to the results of precision measure for different groups of test cases within *Benchmark biblio ontology track.*

CroMatcher with the Autoweight+++ method achieved better results of precision than CroMatcher with the DGWA method for all groups of test cases (No-LC, No-LCS, No-LCI and No-LCP) except for the group ALL where the systems achieved the same results. Hence, CroMatcher with the Autoweight+++ method reduced the number of false correspondences (higher precision value) within the final matching results a little better than CroMatcher with the DGWA method. However, it is not a significant difference in results of precision between two versions of system and the results of F-Measure, which is the most important measure for the evaluation of the matching process, confirm this equality of two versions of systems. The result of F-measure can be seen in Figure 12.

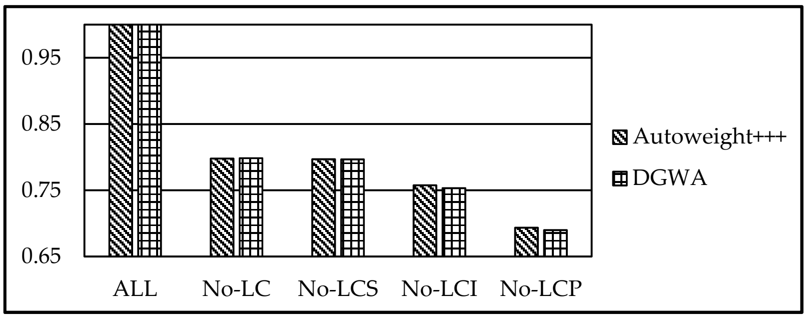


Figure 12. Comparison of methods Autoweight+++ and DGWA according to the results of F-Measure for different groups of test cases within *Benchmark biblio ontology track.*

Here, the CroMatcher with the DGWA method achieved slightly better results of F-Measure than the CroMatcher with the Autoweight+++ method for two groups of test cases (No-LC and No-LCS) while CroMatcher with the Autoweight+++ method achieved slightly better results of F-Measure for No-LCI and No-LCP groups of test cases. The evaluation described above proves the quality of both methods for determining weighted factors. As stated before, although the matching results for the complete *Benchmark biblio ontology track* produced by two versions of CroMatcher are almost equal, the matching results of these two versions of system for individual test cases within the *Benchmark biblio ontology track* are quite different. Therefore, it can be assumed that there is still room for improvement when matching *Benchmark biblio ontology track* which is the reference point for evaluating an ontology matching system.

Furthermore, it is interesting to see the values of weighted factors determined by Autoweight+++ and DGWA methods, while matching the pair of ontologies of the *Benchmark biblio ontology track* with two versions of CroMatcher system. Although the overall matching results are almost equal, the weighted factors of basic matchers determined by the Autoweight+++ and DGWA methods are completely different. As stated before, CroMatcher [1] system contains three weighted aggregations: the aggregation of string basic matchers, the aggregation of structure basic matchers and the aggregation of aggregated results obtained by the first two aggregations. When the aggregation of string basic matchers is performing, the matching results of five basic matchers (Annotation matcher, Profile matcher, Instance matcher, Additional instance matcher and Constraint matcher) must be aggregated. The average values of weighted factors assigned to the string matchers by using methods Autoweight+++ and DGWA, while matching the entire *Benchmark biblio ontology track*, can be seen in Figure 13.

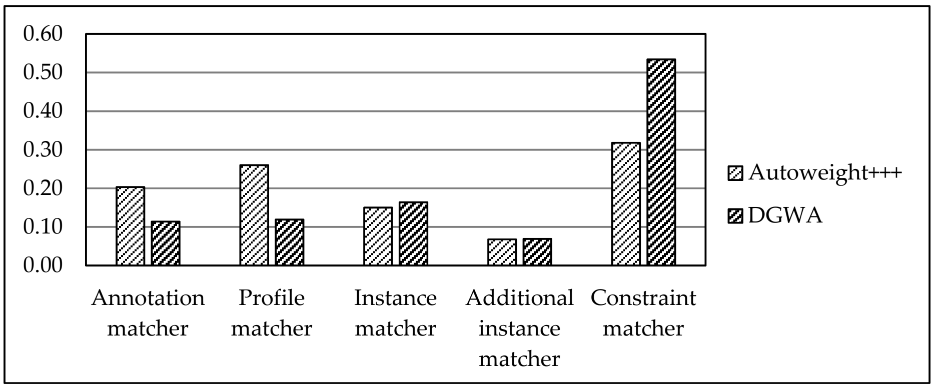


Figure 13. The average values of weighted factors assigned to the string matchers by using aggregation methods Autoweight+++ and DGWA while matching the entire *Benchmark biblio ontology track.*

Both methods determine the biggest weighted factor for the Constraint matcher, but the average value of weighted factor of Constraint matcher determined by DGWA method is almost twice as greater than the average value weighted factor of Constraint matcher determined by Autoweight+++ method. The average values of Annotation and Profile matchers determined by Autoweight+++ method are much higher than average values of these matchers determined by the DGWA method. Hence, the values of weighted factors assigned to the string matchers by using aggregation methods Autoweight+++ and DGWA are quite different. Later in this section, the evaluation results show that, although the number of correct correspondences found by two versions of CroMatcher system are almost equal, CroMatcher system with DGWA method finds more reliable correspondences because these correspondences have much higher confidence value than the correspondences found by CroMatcher system with Autoweight+++ method. As the two versions of systems consists of the same components except the method for determining the weighted factors, it can be concluded that the DGWA method (with its determined weighted factors) influences on the values of correspondences found by matching system. In Figure 14. the average values of weighted factors assigned to the structure matchers by using methods Autoweight+++ and DGWA, while matching the entire *Benchmark biblio ontology track*, are shown. When the aggregation of structure basic matchers is performed within the matching process of CroMatcher system, the matching results of four basic matchers (SuperEntity matcher, SubEntity matcher, Domain matcher and Range matcher) must be aggregated.

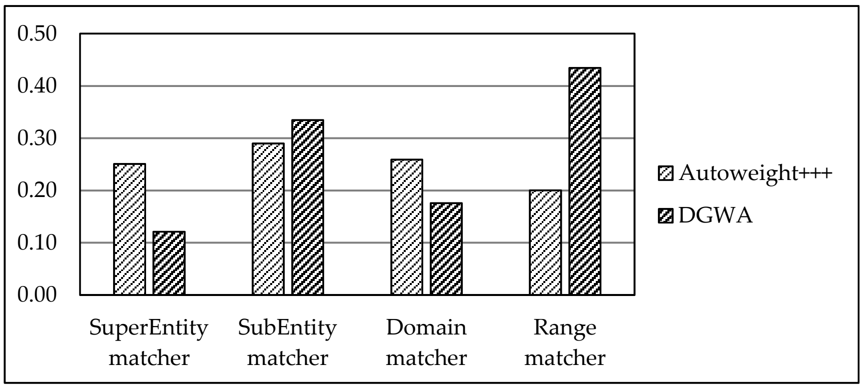


Figure 14. The average values of weighted factors assigned to the structure matchers by using aggregation methods Autoweight+++ and DGWA while matching the entire *Benchmark biblio ontology track*.

The values of weighted factors of structure matchers determined by methods DGWA and Autoweight+++ are completely different. DGWA method determines the biggest weighted factor for the Range matcher, while the Autoweight+++ method determines the biggest weighted factor for the SubEntity matcher. According to the values of weighted factors, DGWA method determines much higher values for the weighted factors of the Range matcher and the SubEntity matcher than for another two matchers. Contrary to DGWA method, all weighted factors of structure matchers determined by Autoweight+++ method have similar values (between 0.2 and 0.3). The last weighted aggregation within CroMatcher system is the aggregation of aggregated results obtained by the first two aggregations. The average values of the weighted factors assigned to the first two aggregations' matching results by using methods Autoweight+++ and DGWA, while matching the entire *Benchmark biblio ontology track*, can be seen in Figure 15.

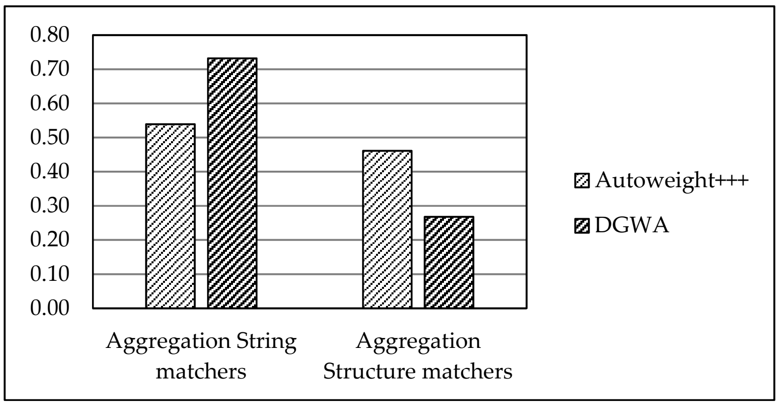


Figure 15. The average values of weighted factors assigned to the aggregation results of string and structure matchers by using aggregation methods Autoweight+++ and DGWA while matching the entire *Benchmark biblio ontology track*.

The weighted factor of the aggregation results of string matchers have much higher value than the weighted factor of the results of aggregation of structure matchers when DGWA method is used in aggregation process. The reverse situation is when the Autoweight+++ method is used in aggregation process. It can be concluded that these two methods produce completely different values of weighted factors of basic matchers. As presented before, the results of recall, precision and F-Measure, which are the most important evaluation measures when matching two ontologies, are almost equal considering two version of CroMatcher system.

However, as stated before, although the matching results (evaluated with recall, precision and F-Measure) produced by two versions of CroMatcher system are almost equal, CroMatcher system with DGWA method finds more reliable correspondences, because these correspondences have much higher value than the correspondences found by CroMatcher system with Autoweight+++ method. The comparison between two versions of CroMatcher system (one with Autoweight+++ method and another with the DGWA method) considering the average values of correctly found correspondences for each test case within *Benchmark biblio ontology track*, can be seen in Figure 16.

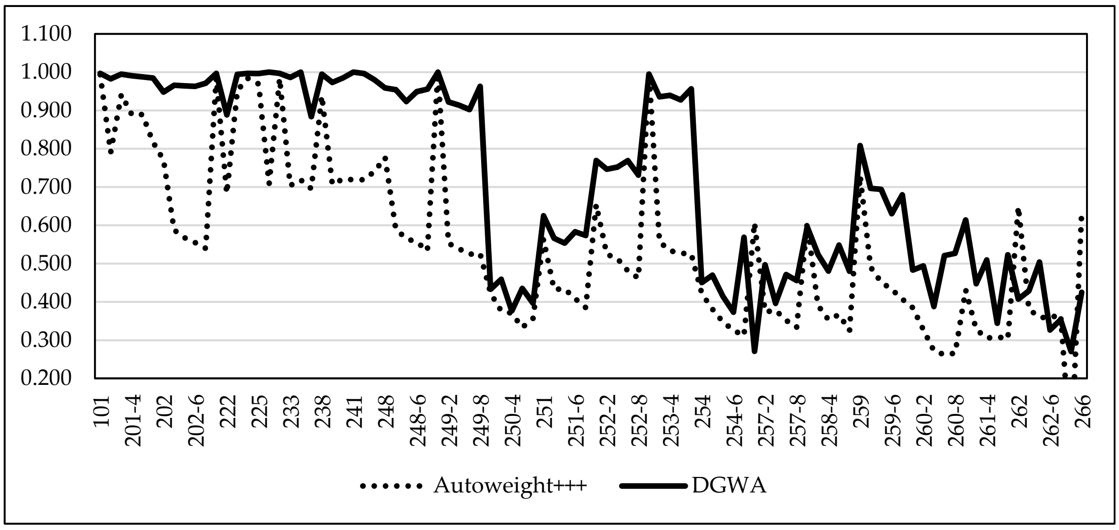


Figure 16. Comparison between two versions of CroMatcher system (one with Autoweight+++ method and another with the DGWA method) considering the average values of correctly found correspondences for each test case within *Benchmark biblio ontology track.*

In almost all test cases within *Benchmark biblio ontology track*, the average values of correctly found correspondences obtained by the version of CroMatcher with DGWA method are much higher than the average values of correctly found correspondences obtained by the version of CroMatcher with Autoweight+++ method. Considering the average value of all correctly found correspondences in all test cases within *Benchmark biblio ontology track*, the value for the CroMatcher with DGWA is 0.72, while the value for the CroMatcher with Autoweight+++ is 0.56. Hence, it can be concluded that CroMatcher with DGWA method determines more reliable correspondences than the CroMatcher with Autoweight+++ method, which was the best method for automatic execution of weighted aggregation process so far [6]. Therefore, the usage of the DGWA method within the CroMatcher system has improved the process of matching two ontologies by increasing the quality of matching results with higher confidence values of correctly found correspondences.

6. Conclusions

Today, many data sources that describe the same domain of interest exist. Although these data sources describe the same domain of interest, they are usually designed independently of each other and thus they are mutually heterogeneous. At some later point, such heterogeneous sources that contain the data of the same domain of interest frequently need to be coupled. An ontology enriches the knowledge of a data source by giving a detailed description of entities and their mutual relations within the domain of interest. The use of ontologies eases the integration of heterogeneous data sources that belong to the same domain.

Ontology matching is the process of finding correspondences between entities of different ontologies, therefore it is a key issue in integrating heterogeneous data sources described by ontologies. When implementing an ontology matching system, it is necessary to automate the matching process to ease the usage of a system for ordinary (i.e., non-expert) users. Thus, the objective of this research was to improve the matching process of automatic matching system without decreasing the quality of the found correspondences between two compared ontologies.

In this paper, we proposed the DGWA method for automatic determination of the weighted factors of basic matchers in the weighted aggregation of these basic matchers by using genetic algorithm. The determination of the weighted factors of basic matchers is a very delicate issue, because due to uniqueness of each ontology, the efficiency of any basic matcher depends on the implementation of the ontologies that take part in the matching process. Hence, the system must recognize which basic matchers produce good matching results for certain pair of currently compared ontologies to determine higher values of weighted factors for these basic matchers and thus to produce better matching results.

Furthermore, we tested DGWA method on *Benchmark biblio ontology track* (the largest test set in the ontology matching evaluation organized by OAEI) by embedding this method within CroMatcher matching system which was one of the best ontology matching system according to the matching results produced for *Benchmark biblio ontology track*. CroMatcher with DGWA method produced excellent matching results for *Benchmark biblio ontology track*. We also made a comparison between two versions of CroMatcher: one with new DGWA method and one with Autoweight+++ method (for automatic determination of the weighted factors), which is a part of the last version of CroMatcher system (tested in the OAEI 2016 evaluation). The overall results of these two versions of the system are almost equal, but there are quite different results for many individual test cases produced by these two versions of system, which proves that there is still room for improvement in matching *Benchmark biblio ontology track*. In opposition to Autoweight+++ method, the main advantage of DGWA method is the reliability of the correct correspondences found during the matching process. The version of CroMatcher with DGWA method has improved the process of matching two ontologies by increasing the confidence values of correctly found correspondences. If the confidence value of a correspondence is high, it is more likely that this correspondence represents the correct correspondence between two entities of compared ontologies. Hence, the quality of the matching process has improved by embedding DGWA method within CroMatcher matching system.

In future work we will try to create an enhanced version of DGWA method. The focus will be on the improvement of the fitness function method within the genetic algorithm to achieve even higher test scores.

Author Contributions: M.G. conceived the idea and designed evaluation procedure; B.V. performed the evaluation; M.P. analyzed the evaluation results; M.G., M.P. and B.V. wrote the paper.

Acknowledgments: This work has been completely supported by University of Rijeka, Faculty of Maritime Studies (Croatia) that provides the funds for article processing charge which are awarded through the Rule book for supporting research activity.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gulić, M.; Vrdoljak, B.; Banek, M. CroMatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment. *J. Web Semant.* **2016**, *41*, 50–71. [CrossRef]
2. Borst, P.; Akkermans, H.; Top, J. Engineering ontologies. *Int. J. Hum.-Comput. Stud.* **1997**, *46*, 365–406. [CrossRef]
3. Antoniou, G.; Harmelen, F.V. Web Ontology Language: OWL. In *Handbook on Ontologies*; Staab, S., Studer, R., Eds.; International Handbooks on Information Systems; Springer: Berlin/Heidelberg, Germany, 2009; pp. 91–111, ISBN 978-3-540-92673-3.
4. Euzenat, J.; Shvaiko, P. *Ontology Matching*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2013; ISBN 3-540-49611-4.
5. Broder, A. A Taxonomy of Web Search. *SIGIR Forum* **2002**, *36*, 3–10. [CrossRef]

6. Gulić, M.; Magdalenić, I.; Vrdoljak, B. Automatically Specifying Parallel Composition of Matchers in Ontology Matching Process. In Proceedings of the Research Conference on Metadata and Semantic Research (MTSR 2011), Izmir, Turkey, 12–14 October 2011; García-Barriocanal, E., Cebeci, Z., Okur, M.C., Öztürk, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 22–33.
7. Mao, M.; Peng, Y.; Spring, M. A Harmony based adaptive ontology mapping approach. In Proceedings of the International Conference on Semantic Web and Web Services (SWWS 2008), Las Vegas, NV, USA, 14–17 July 2008; Arabnia, H.R., Marsh, A., Eds.; CSREA Press: Athens, GA, USA, 2008; pp. 336–342.
8. Mitchell, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, 1st ed.; MIT Press: Cambridge, MA, USA, 1998; ISBN 0-262-63185-7.
9. Feng, X.; Lau, F.C.M.; Gao, D. A New Bio-inspired Approach to the Traveling Salesman Problem. In Proceedings of the International Conference on Complex Sciences, Shanghai, China, 23–25 February 2009; Zhou, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1310–1321.
10. Euzenat, J.; Roşoiu, M.E.; Trojahn, C. Ontology matching benchmarks: Generation, stability, and discriminability. *J. Web Semant.* **2013**, *21*, 30–48. [CrossRef]
11. Gulić, M.; Vrdoljak, B.; Banek, M. CroMatcher—Results for OAEI 2016. In Proceedings of the 11th International Workshop on Ontology Matching Co-Located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, 17–21 October 2016; Shvaiko, P., Euzenat, J., Jiménez-Ruiz, E., Cheatham, M., Hassanzadeh, O., Ryutaro, I., Eds.; CEUR-WS.org: Aachen, Germany, 2016; Volume 1766, pp. 153–160.
12. Gulić, M.; Vrdoljak, B.; Banek, M. CroMatcher—Results for OAEI 2015. In Proceedings of the 10th International Workshop on Ontology Matching Co-Located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, 12 October 2015; Shvaiko, P., Euzenat, J., Jiménez-Ruiz, E., Cheatham, M., Hassanzadeh, O., Eds.; CEUR-WS.org: Aachen, Germany, 2015; Volume 1545, pp. 130–135.
13. Gulić, M.; Vrdoljak, B. CroMatcher—Results for OAEI 2013. In Proceedings of the 8th International Workshop on Ontology Matching Co-Located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21 October 2013; Shvaiko, P., Euzenat, J., Srinivas, K., Mao, M., Jiménez-Ruiz, E., Eds.; CEUR-WS.org: Aachen, Germany, 2013; Volume 1111, pp. 117–122.
14. Euzenat, J.; Meilicke, C.; Stuckenschmidt, H.; Shvaiko, P.; Trojahn, C. Ontology Alignment Evaluation Initiative: Six Years of Experience. In *Journal on Data Semantics XV*; Spaccapietra, S., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6720, pp. 158–192, ISBN 978-3-642-22630-4.
15. Ontology Alignment Evaluation Initiative. Available online: http://oaei.ontologymatching.org/ (accessed on 25 April 2018).
16. Antoniou, G.; Harmelen, F.V. *Semantic Web Primer*, 1st ed.; The MIT Press: Cambridge, MA, USA, 2004; ISBN 0-262-01210-3.
17. Berners-Lee, T.; Jaffe, C.J. World Wide Web Consortium (W3C). Available online: http://www.w3.org/ (accessed on 24 April 2018).
18. Shvaiko, P.; Euzenat, J. Ontology Matching: State of the Art and Future Challenges. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 158–176. [CrossRef]
19. Do, H.-H.; Rahm, E. COMA: A system for flexible combination of schema matching approaches. In Proceedings of the International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China, 20–23 August 2002; pp. 610–621.
20. Aumueller, D.; Do, H.-H.; Massmann, S.; Rahm, E. COMA++—Schema and ontology matching with COMA. In Proceedings of the International Conference on Management of Data (SIGMOD 2005), Baltimore, MD, USA, 14–16 June 2005; ACM: New York, NY, USA, 2005; p. 906.
21. Ngo, D.H.; Bellahsene, Z. Overview of YAM++—(Not) Yet Another Matcher for ontology alignment task. *J. Web Semant.* **2016**, *41*, 30–49. [CrossRef]
22. Ehrig, M.; Sure, Y. Ontology mapping—An integrated approach. *Semant. Web Res. Appl.* **2004**, 76–91. [CrossRef]
23. Jian, N.; Hu, W.; Cheng, G.; Qu, Y. Falcon-AO: Aligning ontologies with falcon. In Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, AB, Canada, 2 October 2005; Ashpole, B., Ehrig, M., Euzenat, J., Stuckenschmidt, H., Eds.; CEUR-WS.org: Aachen, Germany, 2005; Volume 156, pp. 85–91.

24. Castano, S.; Ferrara, A.; Montanelli, S. Matching ontologies in open networked systems: Techniques and applications. In *Journal on Data Semantics V*; Spaccapietra, S., Atzeni, P., Chu, W.W., Catarci, T., Sycara, K.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3870, pp. 25–63, ISBN 0302-9743.
25. Bach, T.L.; Dieng-Kuntz, R.; Gandon, F. On ontology matching problems (for building a corporate semantic web in a multi-communities organization). In Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal, 14–17 April 2004; Seruca, I., Cordeiro, J., Hammoudi, S., Filipe, J., Eds.; Springer: Dodrecht, The Netherlands, 2004; pp. 236–243.
26. Dang, T.T.; Gabriel, A.; Hertling, S.; Roskosch, P.; Wlotzka, M.; Zilke, J.R.; Janssen, F.; Paulheim, H. HotMatch results for OAEI 2012. In Proceedings of the 7th International Workshop on Ontology Matching (OM-2012) Collocated with the 11th International Semantic Web Conference (ISWC-2012), Boston, MA, USA, 11 November 2012; Shvaiko, P., Euzenat, J., Mao, M., Noy, N., Stuckenschmidt, H., Eds.; CEUR-WS.org: Aachen, Germany, 2012; Volume 946, pp. 145–151.
27. Wang, P.; Wang, W. Lily results for OAEI 2016. In Proceedings of the 11th International Workshop on Ontology Matching Co-Located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, 17–21 October 2016; Shvaiko, P., Euzenat, J., Jiménez-Ruiz, E., Cheatham, M., Hassanzadeh, O., Ryutaro, I., Eds.; CEUR-WS.org: Aachen, Germany, 2016; Volume 1766, pp. 178–184.
28. Zhang, Y.; Wang, X.; He, S.; Liu, K.; Zhao, J.; Lv, X. IAMA results for OAEI 2013. In Proceedings of the 8th International Workshop on Ontology Matching Co-Located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21 October 2013; Shvaiko, P., Euzenat, J., Srinivas, K., Mao, M., Jiménez-Ruiz, E., Eds.; CEUR-WS.org: Aachen, Germany, 2013; pp. 123–130.
29. Gracia, J.; Asooja, K. Monolingual and cross-lingual ontology matching with CIDER-CL: Evaluation report for OAEI 2013. In Proceedings of the 8th International Workshop on Ontology Matching Co-Located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21 October 2013; Shvaiko, P., Euzenat, J., Srinivas, K., Mao, M., Jiménez-Ruiz, E., Eds.; CEUR-WS.org: Aachen, Germany, 2013; Volume 1111, pp. 109–116.
30. Smith, M. *Neural Networks for Statistical Modeling*, 1st ed.; John Wiley & Sons, Inc.: New York, NY, USA, 1993; ISBN 0-442-01310-8.
31. Kuo, I.-H.; Wu, T.-T. ODGOMS—Results for OAEI 2013. In Proceedings of the 8th International Workshop on Ontology Matching Co-Located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21 October 2013; Shvaiko, P., Euzenat, J., Srinivas, K., Mao, M., Jiménez-Ruiz, E., Eds.; CEUR-WS.org: Aachen, Germany, 2013; pp. 153–160.
32. Mao, M.; Peng, Y.; Spring, M. An adaptive ontology mapping approach with neural network based constraint satisfaction. *J. Web Semant.* **2010**, *8*, 14–25. [CrossRef]
33. Xue, X.; Pan, J.-S. An overview on evolutionary algorithm based ontology matching. *J. Inf. Hiding Multimed. Signal Process.* **2018**, *9*, 75–88.
34. Vázquez Naya, J.M.; Martínez Romero, M.; Loureiro, J.P.; Munteanu, C.R.; Pazos Sierra, A. *Improving Ontology Alignment through Genetic Algorithms*, 1st ed.; IGI Global: Hershey, PA, USA, 2010; ISBN 978-1-61520-893-7.
35. Feng, Y.; Zhao, L.; Yang, J. GATuner: Tuning schema matching systems using genetic algorithms. In Proceedings of the 2nd International Workshop on Database Technology and Applications (DBTA2010), Wuhan, China, 27–28 November 2010; pp. 1–4.
36. Martinez-Gil, J.; Alba, E.; Aldana-Montes, J.F. Optimizing ontology alignments by using genetic algorithms. In Proceedings of the First International Conference on Nature Inspired Reasoning for the Semantic Web, Karlsruhe, Germany, 27 October 2008; Guéret, C., Hitzler, P., Schlobach, S., Eds.; CEUR-WS.org: Aachen, Germany, 2008; Volume 419, pp. 1–15.
37. Nikolov, A.; D'Aquin, M.; Motta, E. *Unsupervised Data Linking Using a Genetic Algorithm*; Knowledge Media Institute: Milton Keynes, UK, 2011; pp. 1–17.
38. Jebari, K.; Madiafi, M. Selection Methods for Genetic Algorithms. *Int. J. Emerg. Sci.* **2013**, *3*, 333–344.
39. OAEI 2016 Campaign—Benchmark Results. Available online: http://oaei.ontologymatching.org/2016/results/benchmarks/index.html (accessed on 25 April 2018).