Information Sciences 342 (2016) 53–80

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Symbiosis of evolutionary and combinatorial ontology mapping approaches

Rana Forsati[a,1,*], Mehrnoush Shamsfard[b]

[a] Department of Computer, College of Mechatronic, Karaj Branch, Islamic Azad University, Alborz, Iran
[b] Natural Language Processing (NLP) Research Lab., Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran

ARTICLE INFO

Article history:
Received 20 May 2013
Revised 9 December 2015
Accepted 6 January 2016
Available online 13 January 2016

Keywords:
Ontology mapping
Harmony search
Semantic web

ABSTRACT

The problem of identifying semantically aligned entities in different ontologies known as *ontology mapping* is an outstanding research area and lies at the heart of many semantic applications. The overarching goal of ontology mapping is to discover a valid and comprehensive mapping with the aim of maximizing the number of reasonable alignments of ontological entities. Recently many efforts to automate the ontology mapping have been carried out, with some problems such as scalability and efficiency still evident. In this paper, ontology mapping in heterogeneous knowledge bases is formalized as an optimization problem, and an efficient method called Harmony Search based Ontology Mapping (HSOMap) is proposed, that effectively finds a near-optimal mapping for two input ontologies. In this approach, we make use of many kinds of rating functions, which are also called base matchers to evaluate the similarity of entities. Each base matcher captures the similarity between entities from a different perspective and is able to exploit the available side information about the entities effectively. Also, a novel weighted harmonic-mean method is proposed to aggregate different metrics into a single similarity metric among all pairs of entities from two ontologies. After obtaining the combined similarity metric between ontological entities, a discrete harmony search algorithm is proposed to extract the best alignment. To demonstrate the merits and advantages of the HSOMap algorithm, we conduct a set of experiments on benchmark data sets and compare its performance to other state-of-the-other methods. Our experimental results demonstrate that applying harmony search in the context of ontology mapping is a feasible approach and improves the mapping effectiveness significantly.

1. Introduction

The emergence of intelligent information systems resulted in the need for information and knowledge to be represented in a readable form for machines. Nowadays, semantic web technologies are bringing about a revolutionary change on the Internet where the World Wide Web will not just be based on text-based pages, but those with meaning (semantics) attached

* Corresponding author.
E-mail addresses: forsati@cse.msu.edu (R. Forsati), m_shams@sbu.ac.ir (M. Shamsfard).

[1] Most of this work was done when this author was a Ph.D. student at Natural Language Processing (NLP) Research Lab., Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran.

http://dx.doi.org/10.1016/j.ins.2016.01.025
0020-0255/

to them. As the result, ontologies have become the backbone of facilitating the fulfillment of the semantic web vision. However, in recent years number of ontologies have been developed with different terms or taxonomies, where many of them describe similar domains and contain overlapping information. The diversity of available ontologies leads to the problem of ontology heterogeneity [31], which in turn leads to challenging problems in knowledge sharing.

Ontologies in the context of computer science have been formally described by Gruber [27] in 1995 as "*an explicit formal specification of a conceptualization*"— a description which has frequently been refined and reinterpreted. Applications may require using ontologies from different areas or from different views on one area. Moreover, ontology developers may need to use existing ontologies as the basis for the creation of new ontologies by combining knowledge from different smaller ontologies or extending accessible ontologies. In each of these cases we need to find the relationships between the entities in the different ontologies. To this end, scientists came up with the Ontology Mapping (OM) problem (also called aligning or matching). The overarching goal of the OM is to discover alignments among the semantically related entities of different ontologies. The OM problem has emerged as a crucial step when information sources are being integrated in order to be used for various tasks, such as ontology merging, data translation, query answering or navigation on the web of data [48]. As a result, ontology mapping can be considered as one of the key technologies for efficient knowledge exchange and successful understanding of semantic web. This is precisely the problem that has been addressed in this paper. So far, even though researchers have been working on ontology mapping for quite a long time and have solved some conflicts, developing high quality mapping systems is still a challenging issue.

In general, finding a unique best alignment among ontologies is hard or even impossible to accomplish where ontologies with tens of thousands of concepts are common. This is due to the fact that ontology mapping suffers from exponential running time that limits its applicability (except for small ontologies). More precisely, mapping two ontologies in polynomial running time is impossible as the problem is proven to be MAX SNP-hard [3]. To alleviate this computational burden, a variety of heuristic methods for automatic mapping of ontologies have been proposed that differ on their internal model or processing of the required information. These different techniques can broadly be classified into four major categories: string similarity-based, background knowledge-based, structure-based, and combinatorial approaches [18]. The main difficulty in applying these methods is that the searching process is computationally burdensome due to the combinatorial nature of the problem. Therefore, devising efficient searching methods will significantly improve the performance of the mapping process.

Evolutionary algorithms such as genetic algorithm (GA) [40,45,55] and particle swarm optimization (PSO) [4] are high-level general procedures that coordinate simple heuristics and rules to find good approximate solutions for computationally difficult combinatorial optimization problems. Interestingly, evolutionary algorithms (EAs) can be used to perform the search for finding the best matching more efficiently and in a reasonable amount of time. In the context of ontology mapping, results of [4] have identified EAs as a promising technique and have shown that the EAs are the most reasonable methods that are able to obtain higher accuracy compared to other approaches as well as improve the robustness of the typical algorithms used for the same purpose.

The non-exhaustive nature of the search performed by EAs makes them suitable to be combined with combinatorial techniques in ontology mapping. The evolutionary methods provide at the same time a reasonable accuracy, as well as a unique scheme of algorithm when applied to different problems. The symbiosis between EAs and combinatorial ontology mapping is introduced in this paper by using a natural symbiosis of combinatorial ontology mapping techniques and harmony search (HS) algorithm [20,22–24,28,42,56]. The HS algorithm is a meta-heuristic optimization method imitating the music improvisation process, where musicians improvise the pitch of their instruments searching for a perfect state of harmony. Since its inception, over the last few years, the HS algorithm has been vigorously applied to tackle the practical optimization problems in discrete or continuous spaces [19,28,50,57].

A remarkable strength of the HS algorithm hinges on its capability in achieving a good trade-off between exploration and exploitation that makes it more appropriate for optimization problems with complex solution spaces (e.g., combinatorial) such as ontology mapping. Additionally, in contrast to single-point search-based algorithms such as GA in which a unique solution is generated at each iteration, the HS algorithm maintains a set of solutions in harmony memory which evolve at each iteration. Therefore, HS provides an efficient and natural way for exploring the search space and obtaining an acceptable solution. However, when applied to the ontology mapping problem, there are a few issues that require careful consideration including: (i) how to formalize the problem so that it can capture different kinds of mapping cardinalities, (ii) how to take into account the discrete nature of the OM problem in exploring the solution space, and (iii) how to make an efficient implementation of the system to ensure its scalability to large ontologies. The overarching goal of this paper is to address these issues by presenting an efficient algorithm based on the symbiosis between HS and combinatorial mapping methods.

Summary of contributions. The present paper appears to be the first to investigate the symbiosis between HS and combinatorial methods for the ontology mapping problem. In particular, we make the following key contributions:

- We formalize the ontology mapping problem as an optimization problem. Specifically, discovery of mapping between two ontologies is cast as finding the matching with the highest global fitness value rooted in a properly designated objective function.
- We exploit a combinatorial technique to extract the similarity matrix from different similarity measures. A novel parameterized weighted harmonic-mean method to aggregate different similarities into a single similarity is also proposed. The extracted similarity matrix approaches as an appropriate fitness function to guide the search process in the HS algorithm.

- We propose a novel framework called Harmony Search based Ontology Mapping (HSOMap) to conduct ontology mapping by running several processing passes: (i) multi-strategy execution in which each decision independently finds the mapping; (ii) strategy combination in which the mappings of the independent decisions are combined; and (iii) mapping discovery in which some mechanisms are used to discover the best mapping based on the combined rules.
- We also conduct an exhaustive empirical evaluation of the proposed framework and investigate its performance on different benchmark ontologies. In particular, we show that the proposed HSOMap method can be used to efficiently solve the OM on large ontologies.

Outline. This paper is organized as follows. In Section 2 we survey the existing methods. Section 3 provides a formal definition of the ontology mapping problem along with the explanation of the HS algorithm- particularly the aspects necessary to understand our ontology mapping approach. Section 4 provides a detailed description of the proposed HSOMap algorithm. Section 5 discusses the different similarity measures that will be used as the basis for computing the aggregated similarity between concepts. In Section 6 we formally analyze the time complexity of the proposed algorithm. Section 7 presents the data sets used in our experiments, an empirical study of the effects of the HS parameters on the convergence behavior of the HSOMap algorithm, and the performance evaluation of the proposed algorithm compared to other baseline algorithms. Finally, Section 8 summarizes the main conclusions of this work and discusses few directions as future work.

2. More related work

Before discussing the detailed description of the HSOMap algorithm, we would like to draw connections to and put our work in context with some of the more recent work on automatic ontology mapping. A comprehensive overview of the state-of-the-art work in OM is given in [2,17].

Many successful algorithms have been developed over the past few years to map different semantically related ontologies. The different approaches can be roughly divided into the following categories.

String similarity methods. The classical and primary techniques for ontology mapping are based on the string similarity method. The basis of these methods is to calculate the string distance metrics between the labels of two concepts or entities of the different ontologies [13,32,47,51,52]. A comparison of different mapping methods based on string similarity from distance functions to token-based functions can be found in [9]. Some examples of string-based methods which are extensively used in mapping systems are prefix, suffix, edit distance, and n-gram [51].

Background knowledge based approaches. Another family of mapping methods involve utilization of background knowledge or synonyms about ontologies. In these methods, usually one or more linguistic resources in the form of a lexicon, dictionary, thesaurus or some standard reference document for a particular domain for identifying the synonym entities is used. These approaches usually use a common knowledge or a domain-specific thesaurus to match words based on their linguistic relations [1]. Some approaches use common knowledge thesauri to obtain the meanings of terms used in ontologies [7,25,44]. Other approaches use domain-specific thesauri, which usually store some specific domain knowledge. We note that this kind of data as well as entries with synonym, hypernym and other relations are not available in the common knowledge thesauri (e.g., proper names) [37].

Structure-based methods. A third approach is based on the structure of the ontologies to be mapped, i.e., the focus is on "Is-a" hierarchy, sibling concept, relation and graph nodes. The main idea is that two entities of the source and target ontologies are similar if they have the same neighbors (structures) and the same attributes [30,32]. This means that the similarity of the nodes as well as the neighboring ones are of concern.

Combinatorial mapping approaches. The combinatorial mapping approaches are based on a combination of two or more of the above-mentioned approaches, which take into account the different aspects of the ontologies in order to obtain better results [1,2]. So far, the achievements of (semi) automatic ontology mapping have been very limited. There are many challenges in this field that need to be resolved to achieve further progress [11].

Evolutionary algorithms for OM. In [40] a method based on the GA is used to determine the optimal weight configuration for a weighted average aggregation of various base matchers which is called Genetics for Ontology Alignments (GOAL). GOAL uses the genetic algorithm in a meta-matching context for optimizing a global alignment function. GOAL does not directly treat the ontology alignment problem as an optimization problem, but rather serves as a meta optimization. It tries to optimize only a global function. The algorithm can only provide an optimal weight configuration for alignment problems, where the optimal solution is already known, because the algorithm requires a reference alignment in order to evaluate its fitness function. For alignment problems with unknown solutions the system can only serve as a heuristic, if the optimal weight configuration has been determined for a similar problem with a known solution. In contrast to GOAL, the GAOM system [55] deals with the ontology alignment problem as an optimization problem. GAOM utilizes a GA, where each chromosome represents an alignment of two ontologies. Each chromosome is evaluated by a fitness function. In GAOM, the genetic algorithm is not part of a meta-matcher, but it is only used for the optimization of random pairs of entities from the two ontologies that represent alignment proposals. Furthermore, the PSO algorithm has been utilized for ontology mapping in [4], where it is viewed and solved as an optimization problem, to serve; correspondences alignment maximization and specifying an accurate set of correspondences that each particle is responsible for.

Table 1
Summary of notations consistently used in the paper and their meaning.

Symbol	Meaning
$\mathcal{O}_s, e_s$	The source input ontology and an entity from it
n_s	The number of entities in source ontology
$\mathcal{O}_t, e_t$	The target input ontology and an entity from it
n_t	The number of entities in target ontology
$n \triangleq \min(n_s, n_t)$	The number of decision variables in optimization
$\mathcal{M}$	The set of all candidate mappings from $\mathcal{O}_s$ to $\mathcal{O}_t$
$M = (m_1, \ldots, m_n)$	An instance from $\mathcal{M}$ with individual alignments $m_i, i \in \{1, 2, \ldots, n\}$
H	The harmony memory
n_{HM}	The harmony memory size (HMS)
p_{HMCR}	The harmony memory consideration rate (HMCR)
p_{PAR}	The pitch adjusting rate (PAR)
NHV	A solution resulting from improvisation step of HS algorithm
n_{G}	The number of generations (iterations) the optimization proceeds
$\phi : \mathbb{R} \to \mathbb{R}_+$	A penalty function used to penalize invalid mappings
$sim_* : \mathcal{O}_s \times \mathcal{O}_t \to [0, 1]$	A function to measure the similarity between two entities ($*$ indicates the type of measure, i.e., lexical, semantic, or structural)
$\Theta : \mathbb{R}_+^k \times \mathbb{R}_+^k \mapsto \mathbb{R}_+$	The similarity aggregation function used to combine the similarity values extracted from k different measures
$F' : \mathcal{M} \mapsto \mathbb{R}_+$	The fitness function used to evaluate the performance of a specific mapping between ontologies

3. Preliminaries

In this section, we describe our setting and the ontology mapping problem more precisely, providing necessary background and defining notation as needed along with the HS optimization algorithm. Table 1 contains a list of the basic notations we used throughout the paper.

3.1. Problem statement

The objective of ontology mapping is to find semantic alignments between similar elements[2] of two different ontologies [47]. In this paper, we deal with 1: 1 alignment, i.e., for an entity in the source ontology, we only match at most one entity from the target ontology. The semantic correspondence between entities is referred to as " $=''$ relationship and elements are referred to as concepts and properties.

Formally, let us consider a source ontology $\mathcal{O}_s$ formed by n_s entities $\{e_{si}, i = 1, \ldots, n_s\}$ in which e_{si} denotes the ith entity of the source ontology, and a target ontology $\mathcal{O}_t$ formed by n_t entities $\{e_{tj}, j = 1, \ldots, n_t\}$ in which e_{tj} denotes the jth entity of the target ontology. The output of an OM algorithm is an alignment between the entities of two ontologies. We represent the result of a specific mapping method with a set of triplets denoted by

$$M = \{(e_{si}, e_{tj}, s) \in \mathcal{O}_s \times \mathcal{O}_t \times \mathbb{R}_+ : \ e_{si} \text{ is aligned to } e_{tj} \text{ with similarity } s\},$$

where each triplet (e_{si}, e_{tj}, s) indicates that the element e_{si} is corresponding to the element e_{tj}, where e_{si} and e_{tj} are named entities issued from the $\mathcal{O}_s$ and $\mathcal{O}_t$, respectively, and the mapping holds a similarity measure of s, which is typically normalized in the range [0, 1]. This measure indicates that element $e_{si} \in \mathcal{O}_s$ is aligned to entity $e_{tj} \in \mathcal{O}_t$ with the similarity value of s.

Let $\mathcal{M}$ be the set of all candidate mappings from the source ontology $\mathcal{O}_s$ to the target ontology $\mathcal{O}_t$. The goal of OM is to find a mapping $M \in \mathcal{M}$ such that the aligned entities attain the maximum similarity formulated as an objective function over the mapping M as will be detailed in later sections. We note that the size of a candidate mapping, i.e., the number of aligned entities, needs to be maximized simultaneously. The two objectives are aggregated into a single objective function that represents the overall evaluation of a candidate solution. We denote the optimal mapping by $M^* \in \mathcal{M}$ which maximizes the attained similarity between aligned entities and respects the size of the alignment. We note that finding the optimal alignment M^* is a challenging endeavor due to the size of the solution space $\mathcal{M}$ and the presence of two contradictory objectives.

3.2. Preliminary data setup

There are a few preparation steps before deploying the HSOMap algorithm. As in this paper the evaluation of solutions by an objective function is rooted in different similarity measures, we have to prepare such information in advance. First, pre-processing of the input ontologies is done to obtain a list of all simple words contained within them, while stop words are removed. After that, the desired similarity measures according to the combinatorial ontology mapping approach are

[2] In this paper, to facilitate the description, we use elements to denote concepts and properties.

selected. Then, these measures are calculated for the source and target ontologies. If there is any need to reuse some of these similarity measures, e.g., in a different combination, it would be easy to access this information without recalculating the similarities. Therefore, this process is prepared only once for each pair of ontologies and each similarity measure. The output of the mapping execution phase with k different similarity strategies, n_s entities in $\mathcal{O}_s$ and n_t entities in $\mathcal{O}_t$ is a $k \times n_s \times n_t$ cube of predicting values, which is stored for later use in the HSOMap algorithm.

3.3. The harmony search algorithm

One of the popular search methods, which mimics the music improvisation process and is a meta-heuristic optimization method, is Harmony Search (HS) [24]. Recent years have witnessed a flurry of research on HS focused on solving different optimization problems, and an interested reader can refer to [22–24,38] for variants of HS algorithm in literature and to [10] for a simple mathematical analysis of the explorative search behavior of HS. The main steps of the algorithm are as follows: (i) initialize the problem and algorithm parameters; (ii) initialize the harmony memory; (iii) improvise a new harmony; (iv) update the harmony memory; and (v) check the stopping criterion. These steps are described in the next five subsections.

3.3.1. Initialize the problem and algorithm parameters

In Step 1, the optimization problem is specified as follows:

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \le 0, & i = 1,\ldots,m \\ & h_j(\mathbf{x}) = 0, & j = 1,\ldots,p \\ & \mathbf{x}_k \in [l_k, u_k] & k = 1,2,\ldots,n, \end{array} \tag{1}$$

where $f(\mathbf{x})$ is the objective function, m is the number of inequality constraints, p is the number of equality constraints, and n is the number of decision variables. The lower and upper bounds for each decision variable are denoted by l_k and u_k, respectively. The HS parameters are also specified in this step. These are the harmony memory size $n_{\text{HM}} \in \mathbb{N}$, or the number of solution vectors in the harmony memory, the probability of harmony memory considering $p_{\text{HMCR}} \in (0, 1)$, the probability of pitch adjusting process $p_{\text{PAR}} \in (0, 1)$, and the number of improvisations or generations $n_{\text{G}} \in \mathbb{N}$ which can also be used as stopping criterion. We perform a sensitive analysis to decide the value of the parameters p_{HMCR}, p_{PAR}, and n_{HM} used in the HS algorithm. The harmony memory, denoted by $\mathbf{H}$, is a memory location where all the solution vectors (sets of decision variables) are stored. The $\mathbf{H}$ is similar to the genetic pool in the GA.

3.3.2. Initialize the harmony memory

In this step, the $\mathbf{H}$ matrix is filled with as many randomly generated solution vectors as the n_{HM}:

$$\mathbf{H} = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \ldots & \mathbf{x}_{n-1}^1 & \mathbf{x}_n^1 & f(\mathbf{x}^1) \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \ldots & \mathbf{x}_{n-1}^2 & \mathbf{x}_n^2 & f(\mathbf{x}^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_1^{n_{\text{HM}}-1} & \mathbf{x}_2^{n_{\text{HM}}-1} & \ldots & \mathbf{x}_{n-1}^{n_{\text{HM}}-1} & \mathbf{x}_n^{n_{\text{HM}}-1} & f(\mathbf{x}^{n_{\text{HM}}-1}) \\ \mathbf{x}_1^{n_{\text{HM}}} & \mathbf{x}_2^{n_{\text{HM}}} & \ldots & \mathbf{x}_{n-1}^{n_{\text{HM}}} & \mathbf{x}_n^{n_{\text{HM}}} & f(\mathbf{x}^{n_{\text{HM}}}) \end{bmatrix}$$

The initial harmony memory is generated from a uniform distribution in the ranges $[l_i, u_i]$, where $1 \le i \le n$. This is done as follows:

$$\mathbf{x}_i^j = l_i + r \times (u_i - l_i), \quad j = 1,2,\ldots,n_{\text{HM}}$$

where $r \sim U(0, 1)$ and U is a uniform random number generator.

3.3.3. Improvise a new harmony

Generating a new harmony is called improvisation. A new harmony vector (NHV), $\mathbf{x}' = (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_n')$, is generated based on three rules: (i) memory consideration, (ii) pitch adjustment, (iii) random selection. In the memory consideration, the value for a decision variable is randomly chosen from the historical values stored in the $\mathbf{H}$ with the probability of p_{HMCR} which is also called the harmony memory consideration rate (HMCR). Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. This operation uses the pitch adjustment rate (PAR) parameter p_{PAR}, which is the probability of applying the pitch adjustment process. If it happens that the decision variable $\mathbf{x}_i'$ is to be pitch adjusted, its value becomes $\mathbf{x}_i' \leftarrow \mathbf{x}_i' + r \times b$ where b is an arbitrary distance bandwidth that captures the amount of maximum change in pitch adjustment, and $r \sim U(-1, 1)$. We note that the pitch adjusting process in this formulation only applies to continuous valued optimization problems, and it must be adopted appropriately to handle discrete valued optimization problems such as OM, which is the focus of current paper. The variables which are not selected for memory consideration will be randomly chosen from the entire possible range with a probability equal to $(1 - p_{\text{HMCR}})$.

Remark 1. The parameter p_{HMCR}, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the $\mathbf{H}$, while $(1 - p_{\text{HMCR}})$ is the rate of randomly selecting one value from the possible range of values. In other

words, p_{HMCR} determines the rate of exploration and exploitation in the course of the optimization process. A high value for p_{HMCR}, forces the algorithm to mostly stick to the exiting solutions in the **H** (i.e, exploitation) and consequently leading to less exploration of the solution space. On the other hand, by choosing a small value for p_{HMCR} the algorithm performs a random behavior in the solution space (i.e, exploration), hence losing all the information collected during the past rounds which deteriorates the effectiveness of the algorithm.

3.3.4. Update harmony memory

If the new harmony vector, $\mathbf{x}' = (\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n)$, has better fitness function than the worst harmony in the **H**, the new harmony is included in the **H** and the existing worst harmony is excluded from the **H**.

3.3.5. Check stopping criterion

The HS is terminated when the stopping criterion (e.g., maximum number of improvisations n_G) has been met. Otherwise, Steps 3 and 4 are repeated.

In meta-heuristic algorithms diversification refers to a form of randomization in order to explore the search space effectively. In the HS algorithm, diversification is essentially controlled by the pitch adjustment and randomization. These are two subcomponents for diversification, which might be an important factor for the high efficiency of the HS method. The diversification issue is of more importance if we consider the ontology mapping problem which has a more complex solution space and requires an efficient way to effectively explore this solution space. If diversification is strong enough, a great number of zones of the search space may be loosely explored, which will reduce the convergence rate of the algorithm. By contrast, if diversification is kept low in the algorithm design, there is a significant risk of leaving a fraction of the solution space unexplored or even producing far-from-optimal solutions due to trapping in local optima.

4. Symbiotic harmony search and combinatorial ontology mapping

In this section, we introduce the HSOMap algorithm based on the symbiosis of the harmony search optimization algorithm and the combinatorial ontology mapping techniques. In this algorithm, the assessment of the established solutions is rooted in the combinatorial measurements extracted by the combinatorial ontology mapping techniques. Additionally, the essential building blocks of the HS algorithm, including the fitness function and the adjustment of parameters, are modified to be determined by these techniques.

The proposed algorithm uses random selection and HS operations to refine the solutions iteratively to boost the accuracy of mappings. Roughly speaking, the HSOMap approach first measures the similarities of syntactic, structural and linguistic information of ontologies in a vector space model by using combinatorial approaches obtained from external sources. Next, it aggregates different similarities according to their predicted performance and adjusts mapping strategies to increase the performance of the final similarity. Finally, it utilizes the HS algorithm to extract the best solution in the context of ontology mapping.

The architecture of the proposed algorithm is shown in Fig. 1. The HSOMap consists of two main modules: the *combinatorial similarity calculator* and the *HS based mapping extraction*. The HSOMap receives as input two ontologies which are supposed to be mapped. First, the similarity calculation process computes a similarity value between possible pairs of entities, one from each of the two ontologies. The details of this calculation, including the description of the different attributes examined for each pair of entities, are provided in Section 5. This process results in several similarity matrices that contain the similarity scores for each pair of elements in the ontologies. The learnt matrices are passed on to be evaluated by the modified HS optimization algorithm to extract a mapping that maximizes the desired objective function. The main steps are outlined below:

1. **Pre-processing:** This step involves a straightforward algorithm which parses the input ontologies using Jena[3] and extracts their elements to compute similarities. In addition, in this step, the name (labels) of concepts, properties, instances and context structure messages are extracted from two ontologies, and normalized by replacing underscores with spaces, splitting at capital letters, removing stop words, stemming, tokenizing, lower casing, eliminating punctuation marks and hyphens or trailing spaces and tabs.
2. **Calculation of similarities by combinatorial approaches:** In this step the similarity between each pair of the extracted elements is calculated by using different strategies. Each strategy only takes on one similarity aspect of two concepts, such as label similarity, comment similarity, profile similarity and structure similarity to measure the closeness of concepts.
3. **Harmony search-based mapping discovery:** In this module, HS is used to find the best mapping. The algorithm consists of an initialization and many improvisation steps to refine the initial solutions. The pitch adjusting process in the original HS algorithm is also modified according to the requirements of HSOMap. The candidate mapping is optimized by finding the best combinations of mapped elements. Each iteration includes two sub-stages: similarities computation based on multiple strategies and HS-based mapping discovery. The process will be carried out until a stopping condition is met.

[3] http://jena.sourceforge.net/

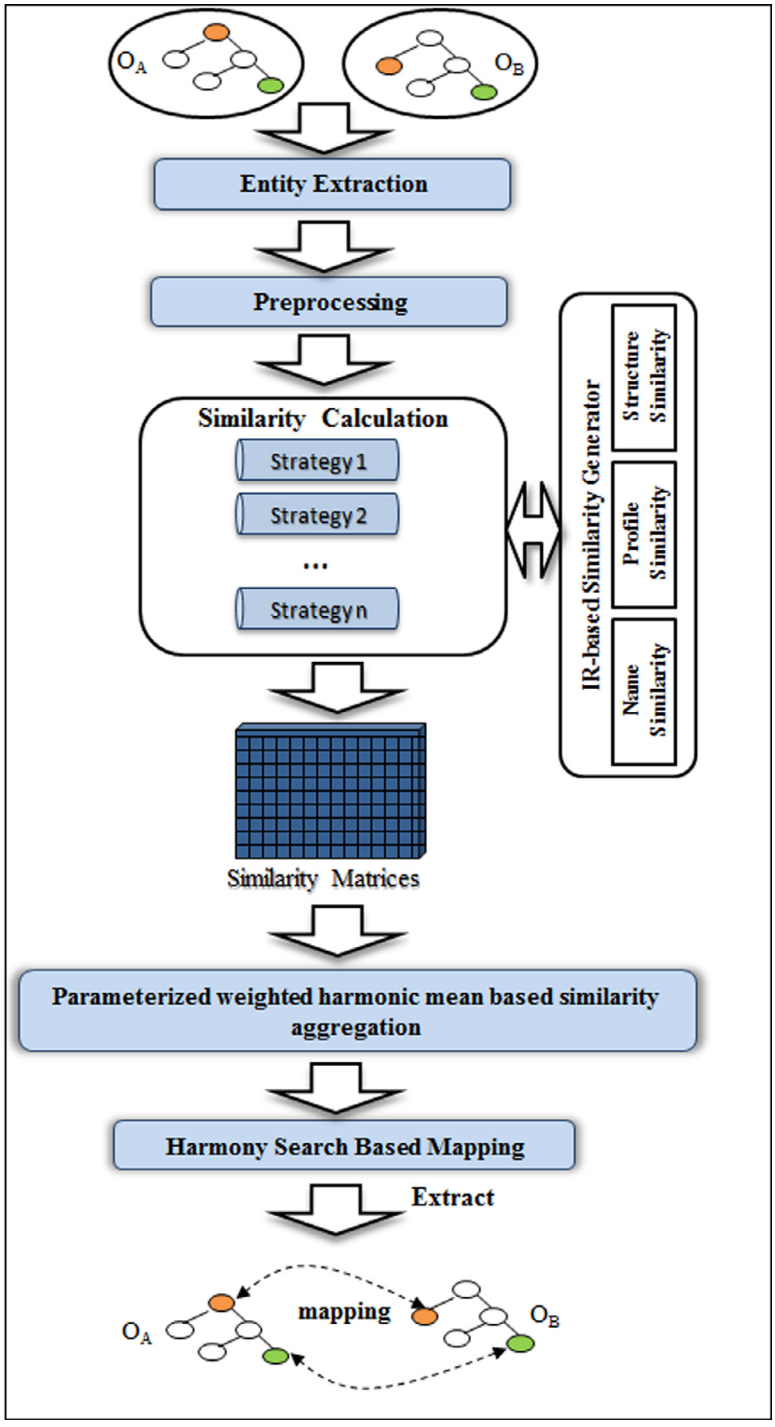

Fig. 1. The architecture of the proposed HSOMap algorithm for ontology mapping with its main modules for finding a near-optimal alignment between two input ontologies.

4. **Post-processing:** The best mapping generated by the HSOMap may or may not satisfy the constraints imposed on the desired solution. We note that different restrictions result in different connections between entities such as the cardinality of a property and no circular mapping. Therefore, post-processing must be conducted on the obtained solution to ensure that it satisfies the imposed restrictions.

The details of these modules, after describing the representation of solutions, are given in the following subsections.

4.1. Representation of solutions and their efficient manipulation

The first decision in solving the ontology mapping problem by the HS optimization method is how to represent solutions in order to apply harmony search operations. In our formulation, each solution in the harmony memory corresponds to a mapping between two ontologies which is encoded as an array of $n \triangleq \min(n_s, n_t)$ elements. To represent this situation we assign to each row of **H** a vector consisting of n integer values, where each position in the vector corresponds to an entity in the source ontology, and its value shows its correspondence chosen among n_t elements from the target ontology. The number of rows denotes the number of different candidate mappings among which we are looking for the best mapping.

To code an ontology mapping, we use hash maps in which keys are entities of the source ontology $\mathcal{O}_s$, and entries are entities of the target ontology $\mathcal{O}_t$. As mentioned before, in this paper only 1: 1 alignments are considered. In other words, each entity $e_{si} \in \mathcal{O}_s$ corresponds to exactly one entity $e_{tj} \in \mathcal{O}_t$ and vice versa. The hash maps structure helps us to easily manipulate a one-to-one mapping between entities, with a search of concepts in $O(1)$ time complexity. Entry for each key is actually the aligned entity of that key in the mapping. Each element in the hash map $\mathbf{H}(e_{si}) = e_{tj}$ indicates that entity e_{si} from the source ontology is aligned to the entity e_{tj} in the target ontology.

We note that the mappings generated in this way may or may not satisfy the constraints that a valid mapping is required to satisfy. Therefore, only the mappings which are consistent with the constraints must be inserted into the harmony memory to ensure the feasibility of solutions in the harmony memory **H**.

4.2. Initialization

The first step in applying the HS algorithm to the problem addressed here is to initialize the harmony memory **H**. For initialization, the harmony memory is filled with as many randomly generated solution vectors as the size of the **H** (i.e., n_{HM}). Each row of the harmony memory is filled by random extraction of one of the elements of the target ontology for each element of source ontology. Although the HSOMap is not sensitive to the initialization of the **H**, an intelligent initialization will slightly decrease the convergence time of the algorithm and improve the global convergence time. Thus, we have improved the HSOMap by proposing two intelligent methods to initialize the **H** with candidate solutions.

The first strategy is a simple random initialization that uses an optional input mapping containing a set of predetermined correspondences to initialize the harmony memory. We note that although the initialization is random, we need to make sure that every entity of the target ontology $\mathcal{O}_t$ appears in the solution once at the most .

In the second strategy, for each entity in the source ontology, the top r ranked corresponding entities in the target ontology $\mathcal{O}_t$ are selected according to the initial similarity measure computed by the lexical similarity between the entities with a predetermined probability of α, and is set to a random entity from the set of entities in the target ontology with a probability of $1 - \alpha$. Positions are filled with random entity index numbers in order to avoid the local minimum solutions, which contributes to maintaining the diversity of the population and helps to generate the new feasible individuals. The corresponding values of those pairs that are not selected, either because their similarity values are lower than the threshold value or they are not in the top r ranked list, will be filled by null. The chosen pairs are further evaluated by other mapping strategies in computing the objective function of solutions. For instance, if for an entity in the source ontology, there is an entity with a similar name in the target ontology, its similarity measure will get a boost, resulting in a change to its rank in the list. There is a parameter which determines the boost size. This process ensures that most of the search space will be explored in initial stages, and in the final stages enough fine tuning will be applied to solutions.

4.3. Reparation of generated solutions

As discussed before, the harmony memory **H** only includes solutions that respect the constraints imposed on the target alignment and represent a valid mapping between the two ontologies. We call a mapping acceptable if it satisfies the following requirements. First, we need to ensure that a solution has a valid structure. A valid structure is an ontology structure that does not have any “Isa” or “part of” loops. Also, as in this paper we only consider 1: 1 mappings; each entity in the source ontology is allowed to be mapped to at most one entity in the target ontology. To ensure these requirements, after the generation of a solution, we need to either modify the generated solution to satisfy the requirements or exclude it from the **H**.

4.4. Improvisation step

In the improvising step, we need a technique to generate a New Harmony Vector (NHV) from all the solution vectors that are in the **H**. The new generated solution must inherit as much information as possible from the solution vectors that are in the **H**. If the generated solution, which corresponds to a new candidate mapping between two ontologies, consists mostly or entirely of all of the assignments found in the vectors in the **H**, it provides good heritability.

The selection of a value for the corresponding element of a source entity is as follows. The index of the corresponding entity of each source entity in the new solution vector is selected with the probability of p_{HMCR} from the harmony memory **H** and with the probability of $(1 - p_{\mathrm{HMCR}})$, which is randomly selected from the set $\{1,2,\ldots,n_t\}$. After generating the new solution, the pitch adjusting process is applied.

We propose a scanning method to generate a new solution. The general mechanism for scanning is to assign a marker to each row in the harmony memory **H** as well as to the NHV. The marker for the new vector traverses through all the positions from left to right, one at a time. In each step, the markers for the rows of the **H** are updated so that at the time of selecting a value for the currently marked element in the new vector, the row markers show the possible choices. The two characterizing features of all scanning procedures are the (row) marker update mechanism and the way in which a value is chosen from the marked elements.

The marker updating mechanism is as follows. The selection of a pair entity from the target ontology for an entity in the source ontology is carried out based on two different strategies. In the first strategy, entities are compared according to their weights. The weight of each element in a mapping is considered as the sum of weights of the pair in which that entity is mapped to. For every entity in the source ontology, e_{si}, all of the entities in the target ontology are examined in the harmony memory, and the best pair, which has the highest weight, is selected with the probability of $p^1_{\text{HMCR}} = (0.1 \times p_{\text{HMCR}})$. In the second strategy, the entity with maximum frequency in the marked positions in the harmony memory is chosen as the best possible entity, with a probability of $p^2_{\text{HMCR}} = (0.2 \times p_{\text{HMCR}})$. Each entity is selected with a probability of p^1_{HMCR} and p^2_{HMCR} from marked elements in the harmony memory. If a matched entity in ontology $\mathcal{O}_t$, e_t is already assigned to another entity in the $\mathcal{O}_s$, then e_t is put in a forbidden list. We note that the random assignment is not done in the middle of an iteration in order to prevent entities of $\mathcal{O}_t$ from being assigned to some random entities that can be assigned to other entities later in the iteration, with better similarity. So this random assignment is postponed until all entities of $\mathcal{O}_s$ are examined for mapping entities in $\mathcal{O}_t$. As an example, assume $e_i \in \mathcal{O}_s$ should be mapped to $e'_i \in \mathcal{O}_t$, but e'_i was previously mapped by some entity from $\mathcal{O}_s$; if at that time we assign e_i to some random entities like $e'_j \in \mathcal{O}_t$, it will avoid a possible good mapping of e_i to e'_i later in the iteration. So this random assignment is postponed until no more assignments are possible. This strategy seems reasonable because the mapping of a single entity in the NHV is no worse than that of the solutions in harmony memory. The pair of a source entity is randomly selected from unselected entities of the target ontology with a probability of $(1 - p_{\text{HMCR}})$.

The value of the ith entity, i.e., $e_i : i = 1, 2, \ldots, n_s$ can be randomly selected from the set of all unselected i $e'_j : j = 1, 2, \ldots, n_t$ with a probability of p_{Random} (i.e., random selection), or it can be selected from the currently marked elements in the **H** with a probability of p_{Memory} (i.e., memory consideration). After generating the new solution, the pitch adjusting process is applied. The pitch adjustment rate p_{PAR} is originally the rate of moving from the current selected entity to a neighboring entity. This parameter in the HS algorithm is a very important parameter in fine-tuning the optimized solution vectors and can be potentially useful in adjusting the convergence rate of the algorithm to the optimal solution.

In contrast to the original HS algorithm and most of its applications, where the HS has been applied to continuous variable optimization problems, our algorithm uses discrete representation of solutions, and we need to modify the pitch adjustment process for this type of optimization. In this computation, p_{PAR} is defined as the rate of moving from one entity to the most probable entity which has the highest value of weight among the other entities. To achieve the best results during the improvising step, we define three different rates for pitch adjusting as $p^1_{\text{PAR}} = (0.6 \times p_{\text{PAR}})$, $p^2_{\text{PAR}} = (0.3 \times p_{\text{PAR}})$, and $p^3_{\text{PAR}} = (0.1 \times p_{\text{PAR}})$. These three rates p^1_{PAR}, p^2_{PAR}, and p^3_{PAR} differ in various iterations and are considered as the rates of moving to the highest similarity score, the second highest similarity score and a previous randomly included entity, respectively. For each entity e_i, whose pair entity is selected from the **H**, with a probability of p^1_{PAR}, the current pair entity of e_i is replaced with a new entity of which e_i has the maximum similarity value to it according to:

$$\text{NHV}[i] = \arg \max_{j \in \{1,2,\ldots,n_t\}} sim(e_i, e_j), \tag{2}$$

where $sim(\cdot, \cdot)$ gives the similarity between two entities from two different ontologies. The similarity between two entities can be computed in different ways as will be discussed later.

According to the above definitions, the current entity is replaced with the entity that has the highest similarity score for that entity in the aggregate weight matrix. For each entity e_i, whose pair entity is selected from the **H**, with a probability of p^2_{PAR}, the current pair entity of e_i is replaced with a new entity of which e_i has the second maximum similarity value to it according to:

$$\text{NHV}[i] = \arg \max_{j \in \{1,2,\ldots,n_t\}, j \neq j_*} sim(e_i, e_j), \tag{3}$$

where $j_* = \arg\max_{j \in \{1,2,\ldots,n_t\}} sim(e_i, e_j)$. For each entity e_i, whose pair entity is selected from the **H**, with a probability of p^3_{PAR}, two random entities whose pair entities are selected from the **H** are selected and their alignments in $\mathcal{O}_t$ are substituted one with the other.

4.5. Evaluation of solutions

The stochastic nature of the technique and the way in which the objective function is converted to a fitness function are mainly considered the main factors that affect the quality of HS solutions. Furthermore, these two factors can guide the mapping process to a desirable part of the search space, leading to high quality solutions. Therefore, we can conclude that the fitness function has a great impact on the outcomes of the algorithm, and it must be designed as accurately as possible.

Let $M = (m_1, m_2, \ldots, m_n)$ represent the set of n alignments for a row in the **H** where m_i is the aligned entity from the target ontology $\mathcal{O}_t$ to the ith entity in the source ontology $\mathcal{O}_s$. Two different methods for fitness evaluation are considered:

The first evaluation method is based on the fitness values of each single alignment in the set of alignments, and the second evaluation strategy is based on the weighted percentage of established alignments between member entities. To this end, rating functions are used for alignment evaluation, which are aggregated to a single fitness value as the similarity of ontological entities. Let $\vec{w} = (w_1, \ldots, w_k)$ be the vector of weights for k similarity measures where w_i reflects the influence of the ith similarity measure. Let $\vec{s_i} = (s_{i,1}, \ldots, s_{i,k})$ be the vector of similarity values assigned by k different measures to the alignment m_i. For each alignment m_i, $1 \le i \le n$, a fitness function $f(m)$ is evaluated as:

$$f(m_i) = \Theta(\vec{s_i}, \vec{w}), \quad 1 \le i \le n, \tag{4}$$

where $\Theta(\cdot, \cdot)$ is a similarity aggregation function detailed in Section 7.1.

Assume $F(M(\mathcal{O}_s, \mathcal{O}_t))$ is a function that returns the fitness of mapping the source ontology $\mathcal{O}_s$ to the target ontology $\mathcal{O}_t$ based on the alignment $M \in \mathcal{M}$. Having computed the different similarity strategies and assigned all possible mapping sequences to the input ontology, our goal is to find a solution that maximizes the objective $F(M(\mathcal{O}_s, \mathcal{O}_t))$ and has the property that, if the $M(\mathcal{O}_s, \mathcal{O}_t)$ is a legal solution with best alignments, it attains the maximum similarity between all mappings of the two ontologies. Also, the level of closeness estimation of an infeasible solution which does not satisfy constraints must be formulated in $F(M(\mathcal{O}_s, \mathcal{O}_t))$. By having this property in $F(M(\mathcal{O}_s, \mathcal{O}_t))$, we are able to lead the convergence to a more stringent space of valid solutions. According to these properties, the fitness of a mapping $M(\mathcal{O}_s, \mathcal{O}_t)$ to evaluate solutions is the average fitness of its correspondences in that mapping solution and is computed by:

$$F(M(O_s, O_t)) = \frac{\mu \times k}{\sum_{i=1}^{n} f(m_i)} \times \prod_{i=1}^{n} \phi(f(m_i) - \Delta_f), \quad m_i \in M, \tag{5}$$

where n denotes the number of alignments in M, Δ_f is a specific similarity threshold, μ is a positive real constant for scaling purposes, $\phi(z)$ is a penalty function defined as follows:

$$\phi(z) = \begin{cases} \frac{rr_D}{z} & if \quad z > 0 \\ 1 - z & if \quad z \le 0 \end{cases} \tag{6}$$

where $r_D < 1$.

Remark 2. It is noted that according to Eqs. (5) and (6), if an alignment has a reasonable similarity value, i.e., $f(m) > \Delta_f$, then the penalty will be small and proportional to how good the alignment is. When the similarity of an alignment is below the Δ_f, the penalty will be larger than 1 and proportional to the dissimilarity of the alignment. The parameter r_D is a known weighting factor, which indicates the highest penalty for a violation. For considering each alignment into safe-set, it must have a higher value than a pre-specified threshold Δ_f.

Typically, by choosing a very small value for Δ_f, the chance of escaping from local optima would be increased, such that r is a self-adaptation penalty factor, which is the function of the current generation as:

$$r_i = r_{min} + \frac{i}{i_{max}} \times (r_{max} - r_{min}) \tag{7}$$

where i_{max} is the total number of iteration, i is the number of current iteration, r_{min} and $r_{max} \in [0, 1]$.

Adaptive penalty functions are favorable due to the fact that they expand the search space and lead to faster convergence by producing low penalty value in the earlier generations, and increasing the penalty value in later generations, respectively. Maintaining diversity and helping to generate the new feasible individual is an objective that can be reached in the initial stage or when the number of iterations is small such that the level of penalty is low. Since distance measures are used to evaluate correspondences, lower evaluation values denote better correspondences mappings. Therefore, if a good result is required, iteration increment cannot be helpful since by doing so, the degree of penalty increases as well, which speeds up the convergence of the algorithm, and the search space may not be searched accurately to find an optimal solution.

As mentioned earlier, another objective in an ontology mapping problem is to maximize the number of alignments in a mapping [4]. Hence, the goal is for all possible candidate mappings $M(\mathcal{O}_s, \mathcal{O}_t)$ to identify:

$$M^* = \min_{M \in \mathcal{M}} F(M(\mathcal{O}_s, \mathcal{O}_t)), \tag{8}$$

which is the mapping M which causes $F(M(\mathcal{O}_s, \mathcal{O}_t))$ to be minimal, and at the same time has the maximum number of alignments.

By aggregating two objective functions into a single fitness value, representing the overall evaluation of the mapping known as a parameterized weighted harmonic, which also respects the size of the mapping.

$$F'((M(\mathcal{O}_s, \mathcal{O}_t))) = \frac{(n - \mid M \mid) \times F(M(\mathcal{O}_s, \mathcal{O}_t))}{\lambda(n - \mid M \mid) + (1 - \lambda)(F(M(\mathcal{O}_s, \mathcal{O}_t))}, \tag{9}$$

where $|M|$ is the number of alignments in solution M.

Maximization of the number of correspondences can be gained in the first part of the sum weighted by λ , through calculating the number of entities of the smaller ontology that are not part of the mapping M, but the evaluation of M can be done in the second part. Hence, the best mapping M^* can be determined according to the following combined objective:

$$M^* = \arg\min_{M \in \mathcal{M}} F'(M(\mathcal{O}_s, \mathcal{O}_t)) \tag{10}$$

Those sequences of mappings with minimum value are considered to be the best outcome of the above equation. In the replacement strategy the newly generated solution is substituted with a row in harmony memory, if the locally optimized vector has a better fitness value than those in the **H**.

4.6. Parameterized harmonic-mean weighted aggregation

The main feature of HSOMap is its capability in utilizing different similarity measures in guiding the search process of HS algorithm. In Section 5 the calculation of lexical, semantic and structural similarities among entities of the source and target ontologies will be discussed. The proposed method produces three matrices for similarity of entities. In order to come up with a single similarity of an alignment, we examine the acquisition of the overall similarity results by combining these matrices- a problem which is referred to as *similarity aggression*.

Similarity aggregation is an important and challenging issue in building ontology mapping systems. In order to come up with a single evaluation of a correspondence, the various similarity distances need to be combined using an aggregation strategy $\Theta : \mathbb{R}_+^k \times \mathbb{R}_+^k \to \mathbb{R}_+$, which takes the similarity values generated by k different measure strategies and the wight of each measure as input and generates a single similarity value. Aggregating different similarities into a single value is pervasive in ontology mapping systems that contain multiple individual matchers [39], e.g., COMA [12], Falcon-AO [46], RiMOM [35,53], and QOM [16], etc. There are many strategies that have been proposed to aggregate different similarities, out of which one can be selected through system configuration.

In this paper a new parameterized harmonic-mean weighted method to aggregate different similarities is proposed. The proposed aggregation strategy computes the flexible merging of all similarity measures via harmonic-mean function. In our proposed evaluation strategy, similarity measures are unequally weighted. The proposed adaptive aggregation method assigns a higher weight to reliable and important similarity measures and a lower weight to those that fail to map similar ontologies. We use a parameterized weighted harmonic mean of similarity measures, which emphasize high individual predicting values and deemphasize low individual predicting values, to represent the combined similarity measure, shown below:

$$\Theta(\vec{s}, \vec{w}) = \frac{\sum_{i=1}^{k} w_i}{\sum_{i=1}^{k} \frac{\omega_i}{s_i}}, \tag{11}$$

where $\vec{s}$ is the similarity values assigned by k different measures to a mapping m, $\vec{w}$ is the weight vector of similarity measures where w_i is the weight assigned to the i^{th} similarity matrix. We use the sigmoid function ϑ: ϑ: [0, 1] → [0, 1] to transform the original similarity values, defined as follows:

$$\vartheta(y) = \frac{1}{1 + e^{-5(y-\nu)}}, \tag{12}$$

where ν is set to 0.5, empirically.

It is widely accepted that weights are useful in such cases to control the contribution of each similarity measure. In our experiments if two given ontologies are more lexically similar than structurally similar, then the lexical coefficient will hold a higher value than the structural coefficient.

5. Similarity computation strategies

The fitness of an extracted mapping is computed from the combination of the fitness values of each single correspondence in the candidate mapping. To this end, each correspondence is evaluated according to some rating functions which are weighted and aggregated to a single fitness value which is used to evaluate the extracted solutions. A rating function is widely known as a base matcher, which computes a single similarity or distance measure of ontological entities.

The HSOMap fitness function is based on the determination of a family of similarity measures which compute a single similarity or distance measure of ontological entities. In fact, the quality of the computed mapping depends to a large extent on the base matchers which are used for evaluating the correspondences. In the evaluation of solutions generated by the HSOMap algorithm, we utilized a *harmonic-mean-weighted* strategy to aggregate multiple similarities along different ontology facets.

This section presents some base matchers, which are currently used as a similarity function. A similarity measure $sim : \mathcal{O}_s \times \mathcal{O}_t \to [0, 1]$ is a function which takes two entities $e_s \in \mathcal{O}_s$ and $e_t \in \mathcal{O}_t$ as input and produces a similarity value between 0 and 1. The similarity between two elements depends on the method used to compute it. It is important for the ease of similarity aggregation that the similarity value is normalized to be between 0 and 1. In the proposed framework, we utilize string-based, semantic-based (i.e., linguistic-based), and structure-based methods to compute the similarities. The string-and linguistic-based methods evaluate the given entities by analyzing their names, labels and comments. They consider both the lexical and linguistic features as terms of comparison. The structure-based method takes into account the structural layout of the ontologies considered.

5.1. Lexical based similarity computation

The lexical similarity consists of all the human-readable information provided in an ontology which refers to different methods of string comparison or edit distances. Three such lexical features are considered in OWL ontologies: the id, the label, and the comment associated with the entities of ontology.

Lexical based similarity for labels and ids. In the first phase of computing the similarity between the entities, we used the most intuitive method which exploits the entity name to compute the similarity of entities. Several approaches have been proposed to conduct the mapping discovery by making use of the entity name [6,14,36]. We used the Levenshtein string-based similarity measure, where the similarity of any two strings (e.g., rdfs:label(s)) from two ontologies is calculated by the Levenshtein distance. The Levenshtein measure defines the strings similarity by the minimum number of insertions, deletions, and substitutions required to transform one string into the other. Since most of the heterogeneity in the test collection ontologies comes from lexical differences, this measure is appropriate for use.

$$sim_{\text{Levenshtein}}(e_s : \mathcal{O}_s, e_t : \mathcal{O}_t) = \max\left(0, \frac{\min(|\ e_s\ |, |\ e_t\ |) - dis(e_s, e_t)}{\min(|\ e_s\ |, |\ e_t\ |)}\right), \tag{13}$$

where $|e_s|$ denotes the length of string e_s, and $dis(e_s, e_t)$ denotes the Levenshtein distance between e_s and e_t.

This similarity measure returns a degree of similarity between 0 and 1. The similarity measure of one would suggest that the two concepts are exactly identical, and a similarity of zero indicates a mismatch. This method considers the number of changes that must be made to change one string into the other and weighs the number of these changes against the length of the shortest string of these two. This will lead to an important question as to whether we will possibly identify the mistakenly identified concepts that have similar spelling, as equivalent. It is entirely possible that such a case might occur, but the probability of two concepts having a very similar spelling and being in the same domain is very low such that the advantage of using it overweighs the risks. The HSOMap finds a lexical similarity measure between identifiers of entities e_s and e_t in the same way as with labels.

Lexical base similarity for profile. Basically, as a collection of weighted words, the virtual vector of an entity in an ontology contains not only the local descriptions (e.g., "rdfs:label(s)") but also some annotation information to reflect the intended meaning of the entity such as "rdfs:comment", and properties and graph structure is exploited to extract the description information from neighboring entities. Then concepts similarities can be calculated by traditional vector space techniques and be further used in similarity based approaches for ontology mapping. The similarity of two vectors in two ontologies based on their local, neighboring and description information, which we called global description of entity, is directly calculated as the cosine measure between their two representative feature vectors. In this model each profile (i.e., label, comment, and properties) [39] is considered as a vector in the term-space. Let χ and ψ denote the number of profiles and the number of terms, respectively. In particular, we have adopted the conventional TF-IDF term weighting model [49] to represent a profile, in which each profile can be represented as

$$\left[tf_1 . \log\left(\frac{\chi}{df_1}\right), tf_2 . \log\left(\frac{\chi}{df_2}\right), \ldots, tf_\psi . \log\left(\frac{\chi}{df_\psi}\right)\right], \tag{14}$$

where tf_i is the frequency of the ith term in the profile and df_i is the number of profiles that contain the ith term. We have used the Cosine similarity to measure the similarity between two profiles as defined by the following equation:

$$sim_{profile}(p_s : O_s, p_t : O_t) = \frac{p_s \cdot p_t}{|p_s| \times |p_t|}, \tag{15}$$

where p_s and p_t are feature vectors for profile p_s and p_t, respectively, and $|p_s|$ and $|p_t|$ are the lengths of the two vectors, respectively.

Lexical similarity measure calculation. The lexical similarity measure is calculated as the weighted average of the label, id, and profile similarities. The weights used in this calculation have been experimentally determined as label weight $w_{label} = 0.5$, id weight $w_{id} = 0.3$ and profile weight $w_{profile} = 0.2$.

5.2. Semantic similarity measure calculation

The next phase in identifying the similarity between the entities of ontologies is to identify entities (concepts) that are semantically equivalent to the concept of the source ontology that is already known to the system. These are concepts or classes that have the same meaning or are equivalent to the concepts already extracted. Lexical based measures ignore the fact that two entity names with similar meanings might be spelt quite differently.

To identify identical concepts we use the assistance of the WordNet lexical database [41] as the knowledge resource for computing the similarity. The urgency of this method is that it is still helpful when the name of elements are different but are linguistically similar [43]. WordNet is a large lexical database organized as a graph. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. This large lexical database can be thought of as serving as an agent that is being used to map the two ontologies, and it helps us to identify similar

concepts. In order to get the meaning of a word in various contexts, synonyms sets (synsets) provided by this dictionary are used. Before using this external linguistic corpora, the entities must undergo a process of linguistic normalization, such as tokenization, lemmatization or word extraction as done in the preprocess phase. Although a wide variety of similarities for WordNet are proposed, after some experimentation we develop a new similarity measure. Assume the two labels being compared are e_s and e_t, belonging to entities, respectively (concepts or properties), the semantic similarity measure between the labels of $e_s \in \mathcal{O}_s$ and $e_t \in \mathcal{O}_t$, is then given as:

$$sim_{\text{Semantic}}(e_s, e_t) = \begin{cases} 1.0 & \text{if} \quad e_s = e_t \\ 0.95 & \text{if} \quad e_s \in sysnset(e_t) \ \text{ or vice versa} \\ 0.0 & \text{if} \quad e_s \in ant(e_t) \\ 0.8 \times \frac{1}{dist(e_s, e_t)} & \text{if} \quad e_s \text{ and } e_t \text{ have a hyponymy-hypernymy relation} \\ 0.7 \times \frac{1}{CommonDepth()} & \text{if} \quad e_s \text{ and } e_t \text{ are siblings with common depth less than 5} \\ \frac{2 \times \log \rho(LCS(e_s, e_t))}{\log \rho(e_s) + \log \rho(e_t)} & \text{o.w.} \end{cases} \tag{16}$$

where $synset(e_t)$ is the set of synonyms and $ant(e_t)$ is the set of antonyms of concept e_t. The similarity measure for synonyms is set slightly lower than the measure for actual string equality matches, in order to differentiate when the terms are exactly matched. $dist(e_s, e_t)$ is the distance between them in WordNet. This can be done by finding the paths from each sense of e_s to each sense of e_t and then selecting the shortest such path, $CommonDepth()$, the common word length, in both descriptions of the words. The final similarity measure synset utilizes the path length of the synset in WordNet. As WordNet is organized with synsets, we can extract the shortest path of the different word pairs using synsets. Synset similarity measures use the path length as the similarity measure. According to [15] we employ the depth and the ancestor least common super concept (LCS) of words. Where LCS is the least common ancestor of e_s and e_t in WordNet, $\rho(LCS) = count(LCS)/total$ is the probability of a randomly selected word occurring in the synset LCS or any sub synsets of it, and total is the number of words in WordNet.

5.3. Structure similarity measure calculation

Structural similarity provides the potential semantic of ontology structure. The RDF model, a foundation of the semantic web, has the nature of a graph structure. Aside from the root entity, in this graph, each element has its corresponding super/sub classes. In this section, we define the similarity measures using the structure of ontologies. For the two entities which came from different ontologies, their matching relationship is bound to be influenced by their super/sub classes. If the super/sub classes can be matched, the same relationship can be found between the two entities of source and target ontology and vice versa. In order to use graphically close concepts, we utilize the parent and child concept label for calculating the similarity. So the structural similarity of two entities $sim_{\text{structure}}(e_s, e_t)$ is defined as:

$$sim_{\text{Structure}}(e_s, e_t) = \frac{sim_{super}(e_s, e_t) + sim_{sub}(e_s, e_t)}{2} \tag{17}$$

where $sim_{super}(e_s, e_t)$ and $sim_{sub}(e_s, e_t)$ is the similarity of their super/sub classes, respectively.

The similarity presented above can handle the similarity of graphical structures. In another view, naturally, two concepts can be regarded as similar if they are domain/range classes of similar properties. The structural similarity between two entities comes from their structural feature which also can be computed as:

$$sim_{\text{Structure}}(e_s, e_t) = \frac{\left| \frac{|e_s|}{|\mathcal{O}_s|} - \frac{|e_t|}{|\mathcal{O}_t|} \right|}{\max\left(\frac{|e_s|}{|\mathcal{O}_s|}, \frac{|e_t|}{|\mathcal{O}_t|} \right)} \tag{18}$$

where $|e|$ denotes the depth of the class e from the root and $|\mathcal{O}|$ denotes the height of the ontology $\mathcal{O}$.

5.4. Property based measure

For each generated concept mapping we make use of property mapping to refine concept mapping in this measure. For each generated concept mapping e_s to e_t, we check mappings of their properties. We give a penalty for those concept mappings when their properties do not have the required level of mappings. We calculate a score that indicates the corresponding percentage of mapped properties in all of their properties. After that, we multiply the combined predicting value of mapping e_s to e_t by the score.

6. Time complexity analysis

In this section the time complexity of the HSOMap algorithm is rigorously determined as a function of the size of the input ontologies. In the first step, for the input ontologies we need to compute k different similarity measures and aggregate them. Then the number of entities involved and the complexity of the respective similarity measures affect the overall runtime complexity of the similarity computation step.

The computational cost of the lexical based measure is dominated by the Levenshtein distance computation module. The Levenshtein or edit distance of two strings can be found using a dynamic programming algorithm. The time complexity of this algorithm is $O(l_1 l_2)$, where l_1 and l_2 are the lengths of the input strings. We note that there are other more efficient algorithms, such as lazy dynamic programming, which are able to compute the edit distance in a more efficient way. Since in our application the length of strings attached to ontological concepts are almost constant, for the simplicity of exposition, we assume that the pairwise similarity between ontological entities has an $O(1)$ time complexity. We note that for other pairwise comparisons of entities the time complexity is also $O(1)$ due to the fact that retrieving single entities or fixed sets of entities is independent of the the size and the structure of the ontology they are taken from. As a result, the total time complexity of computing the similarity matrix based on the lexical information is $O(n^2)$, where n denotes the maximum number of entities in input ontologies.

The time complexity of semantic similarity between two entities is dominated by finding the least common ancestor in the WordNet between any pair of nodes in the ontologies. For an ontology of size n, the depth of a balanced tree over the nodes is at most $O(\log n)$. A naive idea to find the least common ancestor of two entities would be $O(\log^2 n)$, but we made an efficient implementation which only suffers from an $O(\log n)$ time complexity. The idea is to first find the depth of two entities in the tree and move the entity with larger depth up in the tree to reach the same depth as the other entity (this is because the depth of the least common ancestor is less than or equal to the depth of the entity with minimum depth). Then, we can move up in the tree from both entities, one parent at a time, simultaneously, to hit the least common ancestor. The time complexity of finding the depth of entities is $O(\log n)$, and the time complexity of the second step is $O(\log n)$ in the worst case. This leads to an $O(\log n)$ time complexity which is more efficient than the naive $O(\log^2 n)$ complexity for large-scale ontologies. We note that based on the semantic similarity computation detailed before, the process of traversing the tree to find the least common ancestor of each pair of entities must be performed for each pair, which leads to an $O(n^2 \log n)$ time complexity for computing the whole semantic similarity matrix.

The computational complexity of computing the structural similarity between two nodes is $O(1)$ which makes the overall time complexity of computing the structural similarity matrix to be $O(n^2)$. Putting all these together, we can see that the the overall complexity of the similarity computation step is $O(n^2 + n^2 \log n)$. The aggregation of similarity matrices costs $O(kn^2)$. Therefore, the whole process is dominated by the semantic similarity computation step and has an $O(n^2 \log n)$ time complexity.

Remark 3. We note that by examining the semantic similarity computation in 16, it turns out that we only need to check if for any pair of entities, the depth of their corresponding least common ancestor is less than 5. Hence, in the proposed method for finding the least common ancestor, we usually stop the searching process as long as the least common ancestor is found in levels lower than 5. Therefore, in reality the algorithm performs much better than its worst case $O(n^2 \log n)$ time complexity.

We now turn to analyzing the computational complexity of optimization process. The initialization step only takes $O(n^2)$ to initialize the harmony memory. Each improvisation step of the HSOMap requires to generate a new solution. In the worst case, for each entry of the new solution we might apply the pitch adjusting process which takes $O(n)$ operations, as the algorithm needs to find the first and second most similar entities to the current entity. Therefore, the worst case time complexity of the improvisation step gives rises to $O(n^2)$. It is remarkable that the true time complexity of improvisation step is $O(p_{\text{PAR}} \times n^2)$, which is significantly much better than the $O(n^2)$ complexity for the small probabilities of pitch adjusting process. For n_{G} number of improvisation steps, the optimization step costs $O(n_{\text{G}} n^2)$. Combining this with the time complexity of the similarity computation module, the overall time complexity of the HSOMap algorithm becomes $O(n^2(n_{\text{G}} + \log n))$.

7. Experimental results and analysis

In this section, we conduct exhaustive experiments to demonstrate the merits and advantages of the proposed algorithm. We performed experiments with several real world data sets with different characteristics. In particular, we aim to accomplish and answer the following fundamental questions:

1. **Model selection:** What role do the HS parameters p_{HMCR}, p_{PAR}, n_{G}, and n_{HM} play in the performance of the proposed algorithm and balancing the exploration and exploitation? What is the best strategy to tune these parameters?
2. **Alignment accuracy:** How is the quality of obtained solutions on different data sets measured in terms of different quality measures? Additionally, how effective is the proposed fitness function in guiding the search process towards the proximity of the optimal solution in the solution space?
3. **Convergence analysis:** How fast is the proposed algorithm in converging to the best solution?
4. **Effectiveness of similarity aggregation:** To what extent is the proposed aggregation strategy able to combine different similarity measures into a single metric?
5. **Comparison to existing methods:** How well does the HSOMap algorithm perform in comparison to the state-of-the-art methods on different data sets?

We begin by describing the data sets we have used for experiments and then by discussing the results drawn from initial experiments using the HSOMap algorithm on these data sets.

Table 2
The explanation of OAEI benchmark tests.

Data set	Name	Test sets	Descriptions	Number of Ontologies
DS1	1XX	#101–104	Similar both in label description and hierarchy structure	4
DS2	2XX	#201–210	Similar in hierarchy structure but different linguistics in some level	10
DS3	2XX	#221–247	Similar in label description but different structure	18
DS4	2XX	#248–266	Different in both label description and hierarchy structure	15
DS5	3XX	#301–304	Test ontologies are real word ontologies by different institutions	4

7.1. Data sets and evaluation criterias

In order to evaluate our approach, we used the benchmark tests from the Ontology Alignment Evaluation Initiative (OAEI)[4] ontology mapping campaign which become the primary venue for ontology mapping. The OAEI was started in 2004 with the goal of making it easier for researchers to compare the results of their ontology mapping algorithms. The campaign holds a contest every year for evaluating ontology mapping technologies and attracts numerous participants. In addition, the campaign provides uniform test cases for all participants so that the analysis and comparison between different approaches and different systems is practical.

We would like to emphasize that in some cases results published in the contest are only for parts of the ontologies and thus may not indicate true performance for the complete realistic ontologies. Among the 52 ontologies provided in the benchmark data in the domain of bibliography, one is reference ontology, dedicated to the very narrow domain of bibliography, and the rest are test ontologies. The test data are systematically different derivatives of original ontology, which are altered by discarding various information that can be exploited by alignment tools in order to evaluate how an algorithm behaves with missing information. There are different categories of modifications such as removing natural language labels, comments and structural information in order to determine the strengths and weaknesses of different mapping systems. More specifically, the test ontologies in the benchmark data can be classified into five groups, as shown in Table 2. The benchmark test cases have been placed into three categories: 1XX, 2XX, and 3XX test cases. They are grouped into three sets:

1. Concept test (cases 1XX : 101, 102, ...), that explores comparisons between the reference ontology and itself, described with different expressivity levels.
2. Systematic (cases 2XX) that alters systematically the reference ontology to compare different modifications or different missing information.
3. Real ontology (cases 3XX), where comparisons with other real world bibliographic ontologies are explored.

In the OAEI ontology mapping campaign, alignment systems are compared using *precision*, *recall* and *F-measure* metrics [17], which are well-known from information retrieval. Precision is the percentage of correctly discovered alignments in all discovered alignments, and recall is the percentage of correctly discovered alignments in all correct alignments as defined below:

$$\text{Precision} = \frac{\#correct\_found\_mappings}{\#found\_mappings}, \tag{19}$$

$$\text{Recall} = \frac{\#correct\_found\_mappings}{\#existing\_mappings}. \tag{20}$$

The F-measure is the harmonic mean of precision and recall:

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{21}$$

7.2. Baseline algorithms: a comparative analysis

In this section we briefly review the baseline algorithms we intend to compare with the proposed HSOMap algorithm.

1. **MapPSO** (Ontology Mapping by Particle Swarm Optimization) [4,5]: This algorithm has the same spirit as the method proposed in this paper, but it utilizes a discrete particle swarm optimization to solve the ontology mapping problem. Similar to the HSOMap, the core element of MapPSO is the objective function that is used to assess the goodness of candidate solutions. In particular, each candidate alignment of ontologies is scored based on a weighted sum of quality measures of the single correspondences and the total number of correspondences. The quality of each individual correspondence is calculated based on an aggregation of scores from a set of base matchers.

[4] http://oaei.ontologymatching.org/2008/.

2. **Edna:** The Edna is a simple algorithm that is based on the edit distance of the labels and was included by the organizers of the OAEI contest as a baseline. Each label (such as concept name or property name) is composed of several tokens. In this strategy, the edit distance between the labels of two entities is calculated. The edit distance estimates the number of operations needed to convert one string into another.
3. **GeRoMe** (Generic Role based Meta-model) [33]: GeRoMe implements a framework for the management of models that allows operators to manipulate, store, and retrieve models. It is based on the analysis and comparison of five popular meta-models that are widely used in semantic web applications (Relational, EER, UML, OWL, and XML schema). The main idea behind GeRoMe is to decorate each model element (e.g. an XML schema or a relational schema) with a set of role objects that represent specific properties of the model element. In the new representation, each model element plays a set of roles which decorate it with features and act as interfaces to the model element. Roles may be added to or removed from elements at any time, which enables a very flexible and dynamic yet accurate definition of models. The flexibility obtained by this representations makes it possible to manage and manipulate different ontologies in an unified way.
4. **TaxoMap** [29]: The TaxoMap system has been designed in the setting of query answering in the food risk domain. It aimed at increasing answers delivered by a web portal thanks to information provided by other sources annotated by semantic resources. Querying the portal was supported by a global schema exploited by a query interface which had to be reused without any change. More specifically, the TaxoMap is an alignment tool that aims to discover rich correspondences between concepts. It performs an oriented alignment (from the source to a target ontology) and takes into account labels and sub-class descriptions. The target ontology is supposed to be well-structured whereas source ontology can be a flat list of concepts. TaxoMap makes the assumption that most semantic resources are based essentially on classification structures. This assumption is confirmed by large scale ontologies which contain rich lexical information and hierarchical specification without describing specific properties or instances. To find mappings in this context, it only uses the following available elements: the labels of concepts in both ontologies and the structure of the target ontology. Each concept is defined by two elements: a set of labels and subclass relationships. The labels are terms that describe entities in natural language and which can be an expression composed of several words. A subclass relationship establishes links with other concepts.
5. **CIDER** (Context and Inference baseD alignER) [26,54]: The CIDER mapper uses a semantic similarity measure to compare the concepts of the two input ontologies. This schema-based method combines different elementary techniques, such as linguistic similarities or vector space modeling, to compare the ontological context of each of the involved terms. The discovered correspondences that score below a certain threshold are filtered out of the resultant alignment.
   The CIDER algorithm is considered to be a schema-based system (the opposite of others which are instance-based, or mixed), because it relies mostly on schema-level input information for performing ontology mapping. The initial purpose of CIDER was to discover similarities among possible senses of user keywords, in order to integrate them when they were similar enough (to be later disambiguated and used in semantic query construction). The CIDER method compares each pair of ontology terms by, firstly, extracting their ontological contexts up to a certain depth (enriched by using transitive entailment) and, secondly, combining different elementary ontology mapping techniques (e.g., lexical distances and vector space modeling).
6. **SPIDER** [8]: The main goal of the SPIDER system is to provide alignments containing not only equivalence mappings but also a variety of different mapping types (namely, subsumption, disjointedness and named relations). We note that the large majority of existing matching systems focus on deriving equivalence mappings, which distinguishes SPIDER as a different system from existing methods. SPIDER combines two concrete subsystems. First, the CIDER algorithm explained before is used to derive equivalence mappings. Second, this alignment is extended with non-equivalence mappings derived by another system which is referred to as Scarlet. The main focus of SPIDER is on automatic ways of filtering out a significant part of the incorrect mappings.
7. **SAMBO** (System for Aligning and Merging Biomedical Ontologies) [34]: SAMBO is a system for matching and merging biomedical ontologies and assists the user in aligning and merging two biomedical ontologies. It handles ontologies in OWL and outputs 1:1 alignments between concepts and relations. The system uses various similarity-based matchers, including structural, terminological, and background knowledge. The results produced by these matchers are combined based on user-defined weights. Then, filtering based on thresholds is applied to come up with an alignment suggestion, which is further displayed to the user for feedback (approval, rejection or modification). Once matching has been accomplished, the system can merge the matched ontologies, compute the consequences, and check the newly created ontology for consistency, etc.

These algorithms could be compared from different viewpoints including similarity metrics, pre-processing, post-processing, type of alignment, and use of external resources. All these papers were surveyed to determine what lexical metrics were employed and what pre-processing steps were being used (or proposed). For some baseline methods it was not explicitly mentioned which string similarity metrics had been used and we examined the code for the alignment algorithm to extract the metrics to the extent it was possible. The results of this survey are shown in Table 3.

Table 3
A detailed comparison of the proposed method with the baseline algorithms used in our empirical study. Here, for the lexical metrics in the second row, EM stands for exact match, LD stands for Levenshtein distance, LS stands for Lin's similarity, and ED stands for edit distance between two strings. The part-of-speech tagging (PoS-tagging) refers to the problem of assigning syntactic categories to words in a sentence (see e.g., [21]).

Characteristic	Type	TaxoMap	CIDER	SPIDER	GeRoMe	SAMBO	MapPSO	HSOMap
Input format	Ontologies	√	√(OWL or RDF)	√	√	√	√	√
	XML	✗	✗	✗	√	✗	✗	✗
Matching knowledge	Linguistic	√(LS, EM)	√(EM, LD)	√(EM, LD)	√	√(n-gram, ED)	√(SMOA, TF-IDF)	
	Structural	√	√	√	√	√	√	√
	Instance matcher	✗	√	√	√	√	✗	✗
	Use of external dictionary	✗	√(WordNet)	√(Web+ Wikipedia)	√(WordNet)	√(Dictionary+Domain Thesauri)	√(WordNet)	√(WordNet)
Pre-processing	Basic	Stop words, PoS tagging, translation, synonyms	Normalization, synonyms	Normalization, synonyms	Normalization, synonyms	Tokenization, stemming	Normalization	Stop words, PoS tagging, translation, synonyms
Matching techniques		Terminological, structural	Lexical, vector space, structural	Lexical, vector space, domain knowledge	Lexical, structural, role matcher	Terminological, structural, domain knowledge, a learner matcher	Lexical, structural	Lexical, structural

Table 4
Different scenarios considered for evaluating the impact of dynamic parameters on the performance of the HSOMap algorithm. For each scenario, we fix three parameters and only vary the remaining parameter to examine its influence on the quality of the final solution.

Scenario	Fixed parameters	Variable parameter	Values
I	$p_{HMCR} = 0.8,\ p_{PAR} = 0.1,\ n_G = 200$	n_{HM}	1, 10, 20, 30, ..., 150
II	$n_{HM} = 100,\ p_{PAR} = 0.1,\ n_G = 200$	p_{HMCR}	0.03, 0.08, ..., 0.93, 0.98
III	$n_{HM} = 100,\ p_{HMCR} = 0.98,\ n_G = 200$	p_{PAR}	0.05, 0.06, 0.07, 0.08
IV	$n_{HM} = 100,\ p_{HMCR} = 0.98,\ p_{PAR} = 0.05$	n_G	10, 20, 30, ..., 290, 300

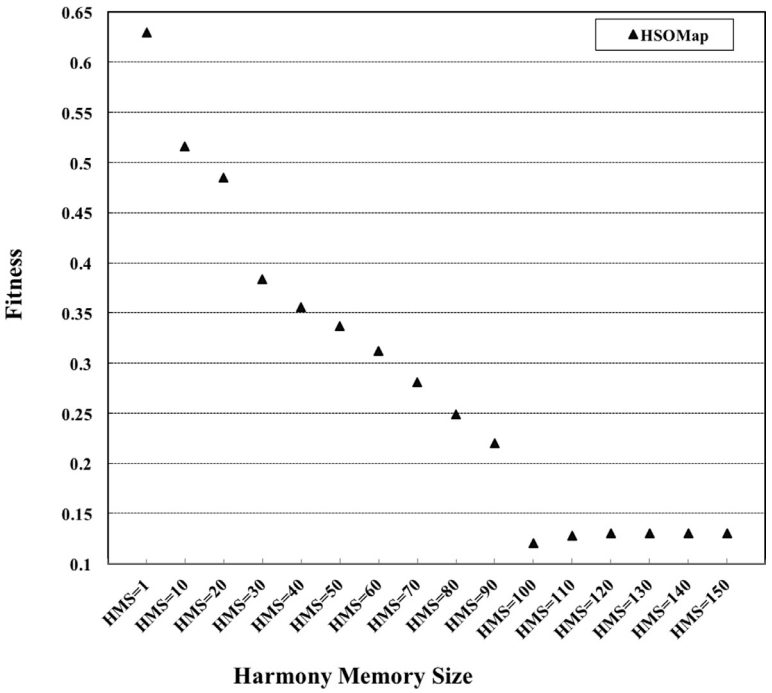

Fig. 2. The performance of the HSOMap algorithm for different values of the harmony memory size (HMS) on a 101 data set corresponding to Scenario (I) in Table 4.

7.3. Empirical study of the impact of dynamic parameters

In this subsection, we investigate the performance of the HSOMap algorithm under different settings of important parameters, i.e., the harmony memory size n_{HM}, harmony memory considering rate p_{HMCR}, pitch adjusting rate p_{PAR}, and the total number of improvisation steps n_G. The goal is to determine the best setting of these parameters for our empirical evaluations to achieve the best performance. To this end, we consider different scenarios where in each of them only one parameter is varied while fixing the values of other parameters. These scenarios are shown in Table 4.

Harmony memory size (n_{HM}). We begin by investigating the impact of the harmony memory size (HMS) on the quality of solutions generated by the HSOMap algorithm. In Fig. 2, we show the results of the first Scenario (I) in Table 4 for the HSOMap algorithm. In this experiment, we fix the values of $p_{HMCR} = 0.8,\ p_{PAR} = 0.1,$ and $n_G = 200$ and change the size of the harmony memory n_{HM} from 1 to 150 with steps of size 10.

Based on the results in Fig. 2, when the size of the harmony memory is $n_{HM} = 100,$ our algorithm achieves the best result, making 100 the most suitable value compared to other values. The results also indicate that the bigger the size of the harmony memory, the better is the chance to start the improving process of the candidate solutions with lower objective values. This might be due to the fact that the large number of solutions in the **H** provides more good shift patterns, which are more likely to be combined into good new solutions.

The other obvious fact that can be inferred from the results in Fig. 2 is that by increasing the size of the harmony memory above 100, performance of the algorithm declines. This may be due to the fact that during the evolution, information of high quality shift patterns have been stored and updated in the **H**. More patterns may contain redundant information, and thus, do not necessarily contribute to a better performance. Therefore, $n_{HM} = 100$ is chosen for all tested instances. Note that when the harmony memory size is $n_{HM} = 1,$ the HSOMap behaves as a local search method, where the p_{HMCR} does not play a role and the pitch adjusting process assists as a local search. The results also indicate that when we vary the size of the

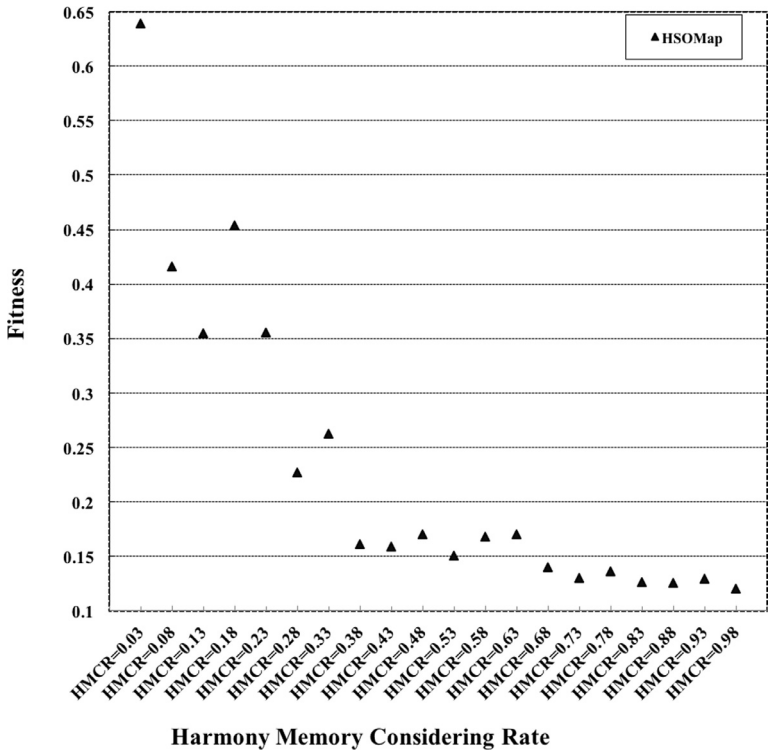

Fig. 3. The performance of the HSOMap algorithm for different values of the harmony memory consideration rate (HMCR) on the 101 data set corresponding to Scenario (II) in Table 4.

harmony memory between 130 and 150, we can not observe any further change in the performance of the algorithm. This phenomena shows that larger values of n_{HM} has a negative impact on the performance and does not necessarily improve the quality of solutions. In particular, for larger values the fitness becomes progressively worse.

Harmony memory consideration rate (p_{HMCR}). In Fig. 3, the second scenario is conducted to investigate the impact of the harmony memory consideration rate (HMCR) on the performance of the HSOMap. As mentioned earlier, the HMCR determines the rate of choosing one value from the historical values stored in the **H**, which is referred to as exploitation. The larger the HMCR is, the less exploration will be achieved and the algorithm further relies on the stored values in the harmony memory, which potentially leads the algorithm to become stuck in a local optimum. On the other hand, choosing an HMCR that is too small will decrease the algorithm efficiency, and the algorithm behaves like a pure random search, with less help from the historical data stored in the memory. As it can be seen in Fig. 3, the fitness value has a downward trend. It means that selecting an HMCR value that is large here could improve the performance, since it relies more on historical data preserved inside the harmony memory. It also indicates that the historical data are effective enough in guiding the process to find the best possible solution. Most of the published HS-based applications such as [19,20] have used a range of values between 0.5 and 0.95 for the HMCR, and our results show that the HSOMap attains the best performance when the HMCR is set to 0.98.

Pitch adjusting rate (p_{PAR}). In the third set of experiments, as shown in Table 4, the HMCR and the HMS are set to 0.98 and 100, respectively, and the fitness of the algorithm for different values of the pitch adjusting rate (PAR) is evaluated. As indicated in Fig. 4, the performance of the HSoMap has not been affected significantly as the PAR values increase or decrease in the specified range. In other words, this interval is reliable enough in terms of selecting different values, resulting in the same or very similar fitness values. We note that for values smaller or larger than the mentioned range, the quality of solutions are significantly worse.

Number of improvisations (n_G). In the last set of the experiments, as shown in Table 4, the suitable value for the number of improvisations is determined. The status of the candidate solutions in the harmony memory **H** during the run is studied to help us to observe the behavior of the **H** and to decide the suitable maximum number of improvisations. It is noted that a significant decrease takes place in the objective function values during the first 150 iterations. Within 150 and 300 iterations, the amount of change becomes very small. After 230 iterations there is no improvement at all in all instances. All that leads us to choose the number of iterations to be 300.

Based on the results of the sensitivity analysis of parameters discussed above, the parameters in the HSOMap algorithm are assigned as follows: the size of the harmony memory is set to 100, the p_{HMCR} and the p_{PAR} probabilities are set to

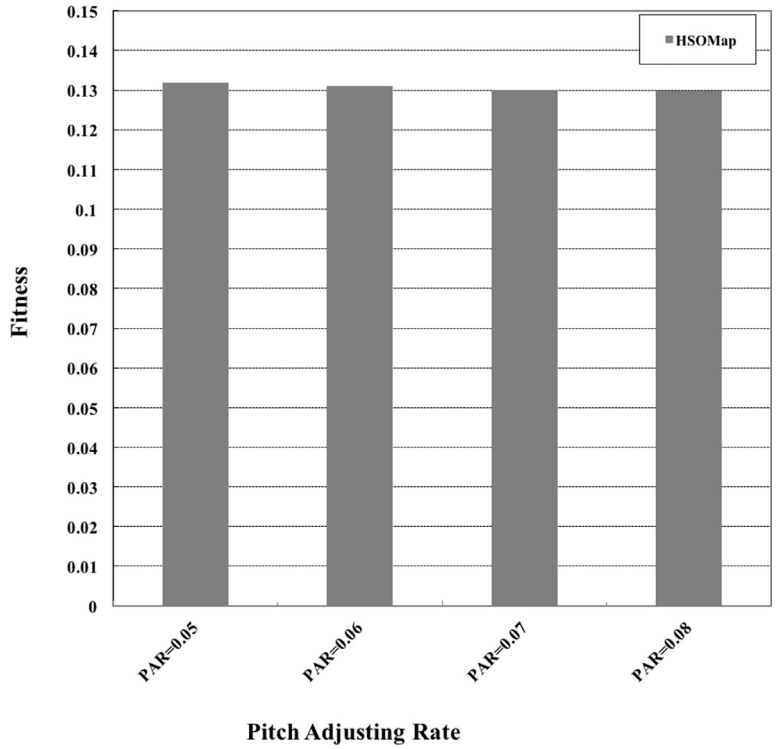

Fig. 4. The performance of the HSOMap algorithm for different values of the pitch adjusting rate (PAR) on the 101 data set corresponding to Scenario (III) in Table 4.

Table 5
The configuration of the parameters for the conducted experiments, obtained by fine-tuning the parameters for different settings.

Parameter	Value
# of improvisations (n_G)	300
Harmony memory size (n_{HM})	100
Harmony memory consideration rate (p_{HMCR})	0.98
Pitch adjusting rate (p_{PAR})	0.05

Table 6
The best achievable fitness value for the HSOMap algorithm on the 101 data set for the different number of improvisation steps corresponding to Scenario (IV) in Table 4.

# of improvisations (n_G)	Fitness
Initial values	0.7650
10	0.7310
50	0.5000
100	0.2840
150	0.0730
200	0.0700
250	0.0698
300	0.0697

0.98 and 0.05, respectively, and the maximum number of iterations is set to 300. The value of dynamic parameters are summarized in Table 5. We note that the HSOMap algorithm is highly adjustable via its parameters and can be tuned to perform well on specific problems in terms of both precision and recall quality measures Table 6.

7.4. The performance of the HSOMap on different data sets

Now we turn to investigating the performance of the proposed algorithm on 1XX, 2XX, and 3XX test cases. The average precision, recall and F-measure values with respect to a given reference alignment obtained for each group using the HSOMap are reported in Fig. 5. The results in Fig. 5 refer to the average alignment found by the HSOMap algorithm after the

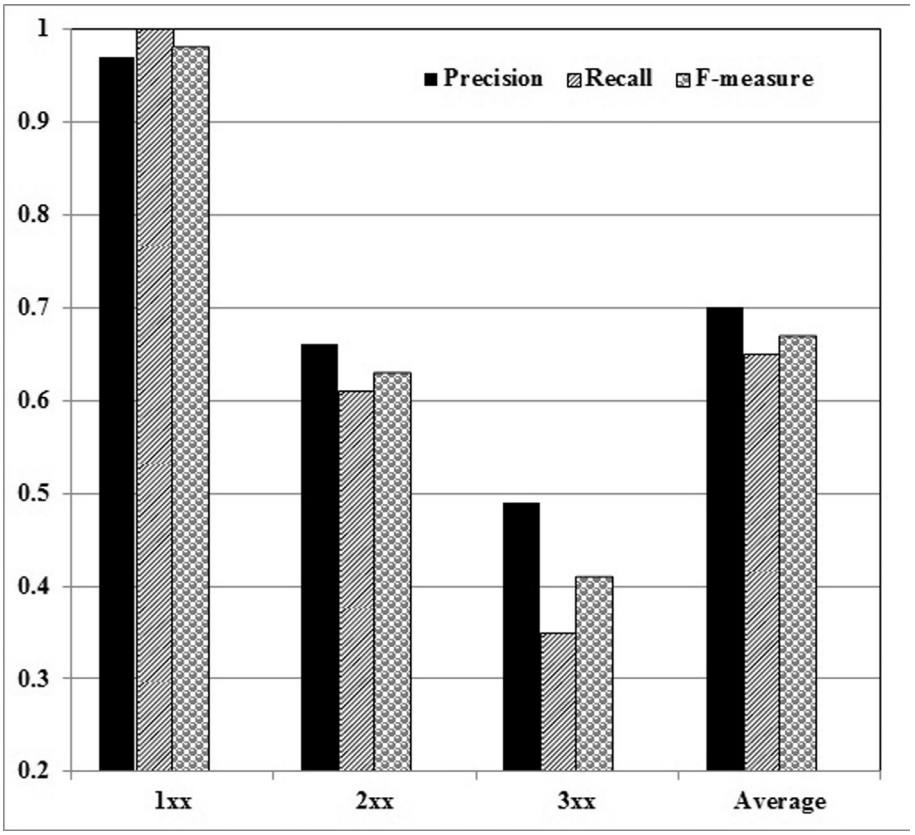

Fig. 5. The performance of the HSOMap on the OAEI-08 benchmark test suite (i.e., 1XX, 2XX, 3XX) measured in terms of precision, recall, and F-measure averaged over independent runs of the algorithm.

given number of iterations. Since the HSOMap is a meta-heuristic search algorithm, it is non-deterministic, and as a result, on a set of independent runs the quality of the results in the alignments will be subject to slight vacillations.

Referring to Fig. 5, we can see that for many of the test cases in the benchmark ontologies, the HSOMap algorithm could provide reasonably good solutions. The HSOMap obtained the best results on test case 1XX. Since there is good lexical and structural information in 1XX, the HSOMap performs perfectly on this test case. This test gets a precision value of around 97% and a recall value of 100%. In test case 2XX the results are not as positive as in test case 1XX. The quality of the established alignments decreases with the decreasing number of features to exploit. Our results show where the lexical and the linguistic information are suppressed from the target ontology, such as in 201 and 202, where the HSOMap has variant behavior. Though the precision and recall results indicate that the alignments extracted by the HSOMap are not completely different from the reference, but they are, in fact, rather close. We have obtained good results in test cases #201–210 and #221–247 in comparison to test cases #248–266. The reason is that in test cases #201–210 and #221–247 the source and target ontologies have similar hierarchy structures and good lexical information, respectively. In test cases #248–266 which have poor lexical and structural information we have acquired the worst results. As the main focus of the HSOMaps is based on lexical and linguistic information, in a situation where the structure information of the target ontologies is changed in test cases #221–247, the results are similar to the 1XX tests. Test case 3XX has four real ontologies, as the results show that the proposed algorithm exhibits a good performance for this test case as well.

7.5. The effectiveness of the aggregation method

The fact that different similarities work well in different situations motivates us to investigate a new measure that can estimate the quality of each similarity so that we can aggregate them according to each one's individual characteristic. We performed experiments to test the effect of our proposed aggregation algorithm. Table 7 shows the results of these experiments. As shown in Table 7, the approach that is used in the aggregation selection can be effective in the performance of ontology mapping, and as it noted, using our proposed aggregation selection method can improve the performance of the HSOMap.

Table 7
The performance of different similarity aggregation strategies in terms of precision and recall.

	HSOMap+ harmonic-mean weighted aggregation		HSOMap+ weighted average aggregation	
Test set	Precision	Recall	Preision.	Recall
1XX	0.97	1	0.95	1
2XX	0.66	0.72	0.66	0.61
3XX	0.71	0.54	0.49	0.35
Average	0.77	0.75	0.70	0.65

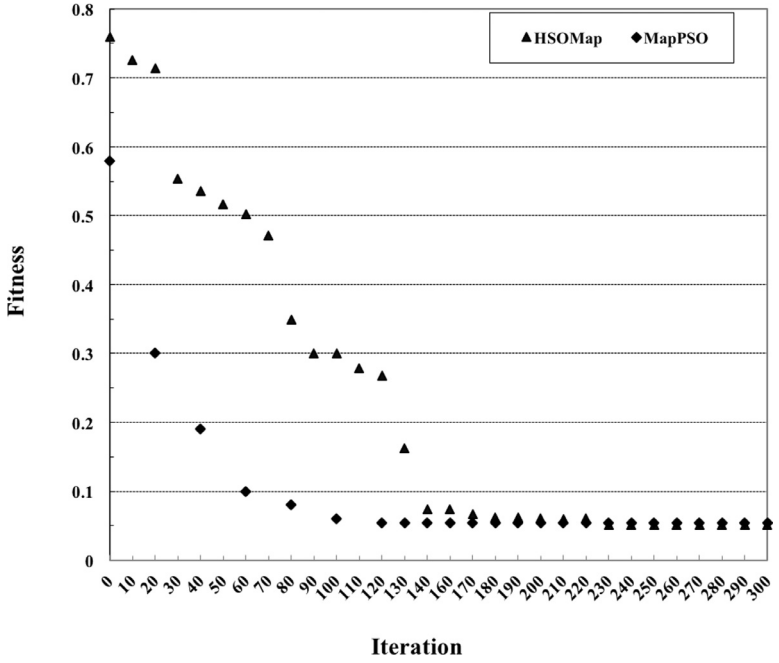

Fig. 6. The convergence rate of the HSOMap and the MapPSO algorithms in test case #101 measured in terms of fitness.

7.6. Comparison with other evolutionary based ontology mapping algorithms

As mentioned earlier, any successful meta-heuristic algorithm requires a good balance between two important, seemingly opposite, components: exploration and exploitation. If the exploitation (a.k.a. intensification) rate is high, only a fraction of solution space might be visited, and there is a risk of being trapped in a local optimum. If the exploration (a.k.a. diversification) rate is set to be high, the algorithm behaves like a random process in the solution space and converges too slowly.

In this part of the experiments, we aim at comparing the behavior of evolutionary based ontology mapping algorithms. To accomplish this goal, two different sets of experiments have been conducted. In the first set, the convergence rate of different evolutionary based ontology mapping algorithms is examined. Then, we investigate how the quality of obtained mapping by different algorithms differ from each other. In relation to other evolutionary based algorithms, the most outstanding tool is the MapPSO algorithm which is a particle swarm based approach for solving the ontology mapping problem [5]. In this algorithm the ontology mapping problem is modeled as a global optimization of a mapping between two ontologies and in this regard is similar to the HSOMap. However, instead of the HS algorithm, the PSO algorithm is used to achieve a quasi-optimal solution. The two algorithms also significantly depart in exploiting side information and modeling the objective function.

In this part of the experiments, the convergence of the evolutionary based algorithms will be investigated. Convergence is the amount of time required by a procedure to reach its best possible solution. The criterion for evaluating the algorithms is their convergence rate to the optimal solution. Fig. 6 illustrates the convergence behaviors of the HSOMap and the MapPSO algorithms on the test case 101 of the OAEI 2008 benchmark track in terms of the fitness function (mapping quality function $F'(M(\mathcal{O}_s, \mathcal{O}_t))$).

[5] Please note that the HSOMap and the MapPSO are not comparable to existing genetic based ontology mappers such as the Genetics for Ontology Alignments (GOAL) algorithm proposed in [40]. The reason is that the GOAL requires a reference alignment in order to evaluate its fitness function which is not applicable to the setting addressed in the paper.

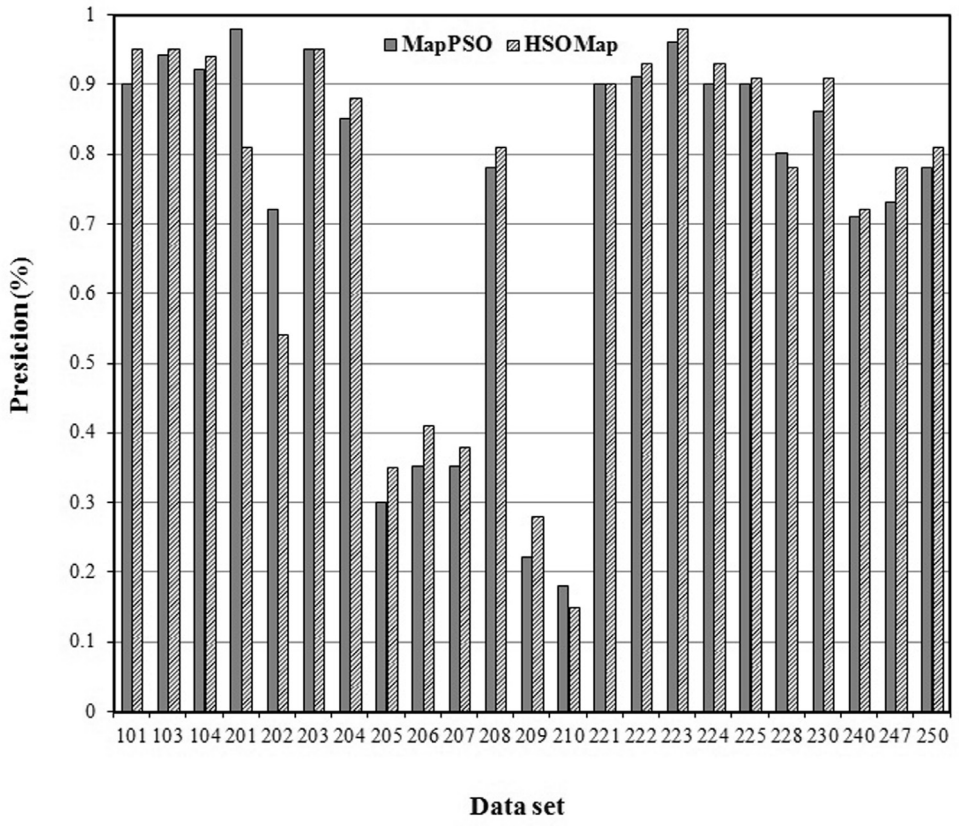

Fig. 7. The average precision of the MapPSO and the proposed HSOMap algorithm on different ontologies.

The horizontal axis shows the iteration number, and the vertical axis corresponds to the fitness value. For this test case we have conducted 10 independent trials with randomly generated initialization, and the average value is recorded to account for the stochastic nature of the algorithm. It can be observed that the mapping quality is continuously improving throughout the iterations in both algorithms. Lower values of the fitness thereby indicate shorter distances and therefore represent better solutions. Comparing the convergence behavior for the HSOMap and the MapPSO demonstrates a slightly different convergence speed for both algorithms, but convergence could be observed for both the HSOMap and the MapPSO.It is obvious from Fig. 6 that the HSOMap converges quickly towards the global minimum. Fig. 6 illustrates that the reduction of the fitness value in the HSOMap follows a smooth curve from its initial vectors to the final optimum solution and does not have a sharp move. The MapPSO demonstrates faster convergence in comparison to the HSOMap, and reaches a stable state after about 120 iterations where no further improvements on the situation occur. In contrast, HSOMap converges more slowly and reaches a stable state after about 150 iterations.

We now turn to comparing the quality of solutions obtained by two methods. The performance comparison between our proposed algorithm with the MapPSO in all data sets, considering precision and recall as quality measures, is demonstrated in Figs. 7 and 8, respectively. It can be inferred from the results of Figs. 7 and 8, the result obtained by the HSOMap is significantly comparable to results obtained by the MapPSO.

From the results of these experiments, one can conclude that although the HSOMap and the MapPSO behave similarly in many cases, in general, results obtained by the HSOMap are comparable to results obtained by the MapPSO. The phenomenon may have its roots in the good exploitative-explorative balance of the HSOMap, the equipped fitness function which measures the similarity from different perspectives, and its parallel nature. Proper parallelism usually leads to better performance with higher efficiency. The good combination of parallelism with elitism as well as a fine balance of exploration and exploitation is the key to the success of the the HSOMap algorithm.

7.7. Comparison to baseline methods

To demonstrate the superiority of the proposed algorithm, it is compared with the baseline algorithms discussed before.

We have compared the HSOMap with CIDER, SAMBO, GeRoMe, MapPSO, SPIDER, TaxoMap, and a few participants of the OAEI-08 benchmark test suite. The results of applying these algorithms to the OAEI-08 benchmark test suite are available in [8]. Note, however, that in some cases results published in the contest are only for parts of the ontologies, and thus,

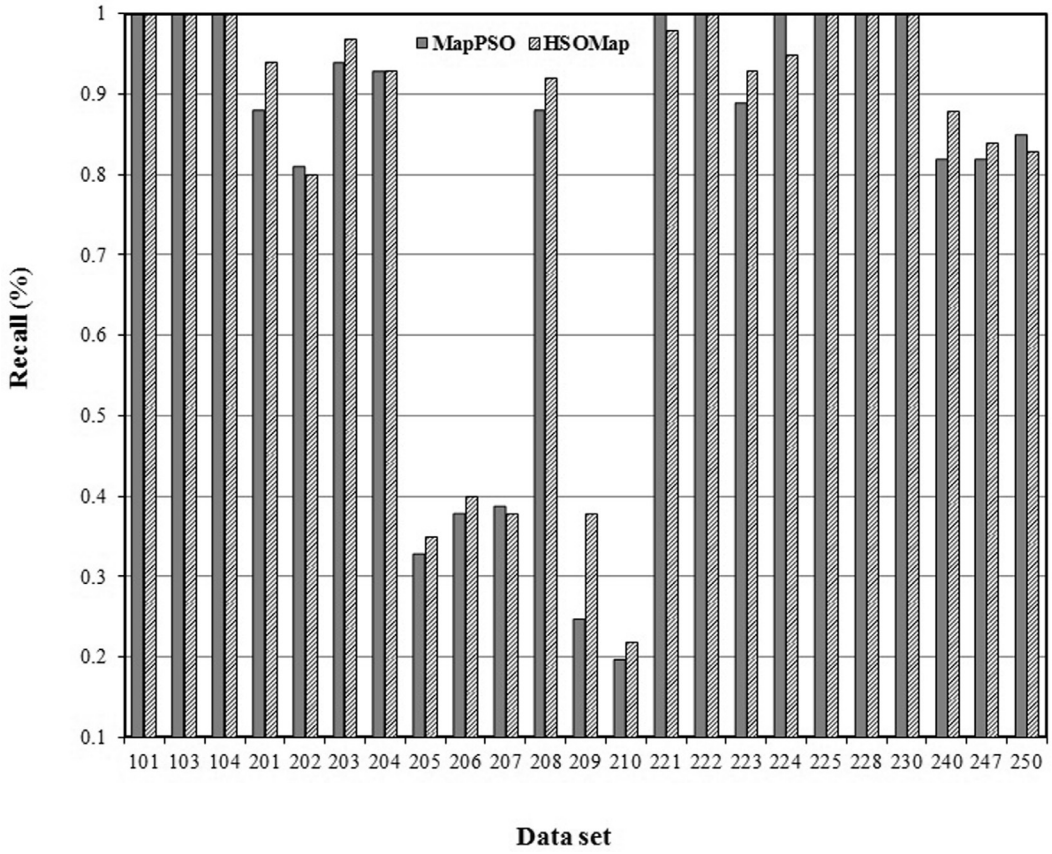

Fig. 8. The average recall of the HSOMap and the MapPSO algorithms on different ontologies.

may not indicate true performance for complete realistic ontologies. The comparison results of the matching quality of the proposed algorithm and the other systems are presented in Figs. 9–11. These figures show the average precision and recall, the harmonic average, and also the F-measure value of these algorithms on three test categories.

Fig. 9 presents the precision of mapping obtained by applying algorithms to different data sets. The most important observation from the experimental results is that the HSOMap was not an overall winner with respect to precision accuracy. Nevertheless, the SAMBO, CIDER and TaxoMap methods, which had better precision than the proposed algorithm, resulted in a lower recall. As an example, the TaxoMap, which managed to reach a good precision in comparison to the HSOMap, had the worst recall and F-measure values among all the algorithms. In fact, it has sacrificed the recall metric for obtaining a better precision.

The performances of the algorithms considering the recall metric as the quality measure is shown in Fig. 10. By comparing the results of different algorithms, it can be seen that the HSOMap has the best performance, which is a considerable improvement in the domain of ontology mapping. The proposed algorithm has focused on achieving the best recall and also managed to reach an adequate precision.

The performance of different algorithms considering the F-measure are shown in Fig. 11. As it can be observed from the results for different algorithms, the HSOMap has a better F-measure value than most of the algorithms which indicates that the proposed algorithm has reached a satisfying efficiency in comparison to theother algorithms. It can be noted from Fig. 11 that the HSOMap outperformed the TaxoMap, MapPSO, Edna, SAMBO and GeRoMe and obtained a very close result to the SPIDER considering the F-measure value.

Putting all the results together, we can conclude that the HSOMap algorithm is able to find a near-optimal solution for ontology mapping in most cases. According to the results, our approach seems to be an accurate and efficient tool for this task.

We now turn to better understand and relate the performance of the different algorithms to their main characteristics outlined in Table 3. We also note that different algorithms perform differently on the test cases which indicates that the type of data set, the richness of side information available in each data set, and the capability of each algorithm in exploiting the information in mapping process significantly affects the performance of each algorithm. Therefore, our discussion is divided in terms of the three test cases.

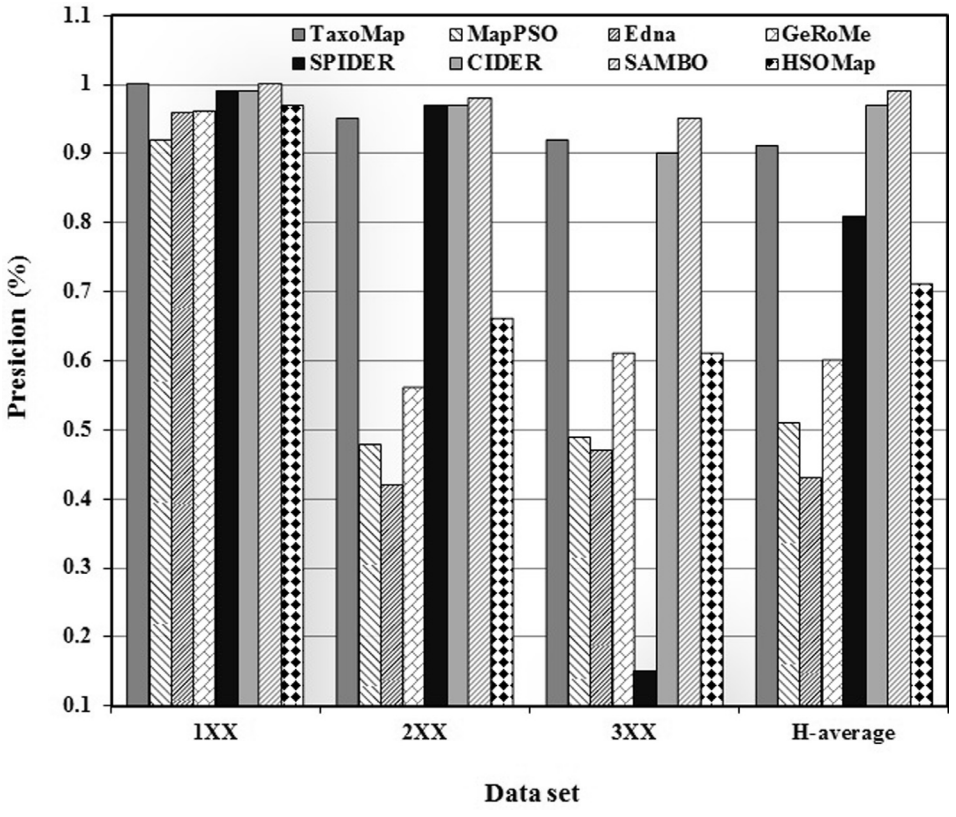

Fig. 9. The average precision of the HSOMap and the baseline methods participated in for the OAEI-08 campaign on 1XX, 2XX, and 3XX test cases.

Test 1XX. Since the TaxoMap algorithm only considers labels and hierarchical relations and only provides mappings for concepts, the recall is low for test case 1XX, even for the reference alignment #101.

The CIDER method considers many features of ontologies and also focuses on the semantic description of the terms in the corresponding ontologies which improve the results in some cases where this information is available. The CIDER obtained a very high precision and recall (≈ 0.99) for this case of tests.

The SPIDER works the same as the CIDER with the difference that it considers non-equivalence mappings. The test set 1XX does not contain non-equivalence mappings in the reference alignment; therefore, it achieves similar results as the CIDER algorithm for this type of tests.

The GeRoMe method achieved a very similar value for precision and recall in comparison to results of other algorithms, where the precision was usually higher than recall. The GeRoMe achieves a high precision and recall for these test cases just by using string matchers.

The MapPSO achieves precision values of around 0.9 and recall values of 1. No system had strictly lower performance than the simple edit distance algorithm on labels (i.e., Edna) in all test cases. Each algorithm has its best score with the 1XX test series. There is no particular order between the two other groups of test cases.

Test 2XX. It is more interesting to look at the 2XX series structure to distinguish the strengths of the different algorithms. These tests are concerned with alternation on labels and hierarchies. More specifically, the group of tests in this category were based on alternations on properties, instances and comments. We note that these alternations do not have any effect on the results obtained by the TaxoMap since it ignores these descriptions. But the TaxoMap for the groups of tests in this category where labels were suppressed or replaced by random string or translated to another language has produced no mapping since the main focus of the TaxoMap is on the linguistic features of the label of concepts. For test case #202 where all names and comments are unavailable, the TaxoMap performs worst in this group of tests. The precision of most remaining tests was very high.

The CIDER algorithm obtained a result with a low recall in this category of experiment. In this category of tests, the CIDER can not provide results for benchmark cases #202 and #248-#266 (in which ontology terms are described with non expressive texts), because the CIDER does not deal with ontologies in which syntax is not significant at all (these cases present a total absence or randomization of labels and comments). Consequently, it has a result with a low recall in this category, as the benchmark test unfavors methods that are not based on graph structure analysis (or similar techniques). It obtained better precision than recall in the benchmark test.

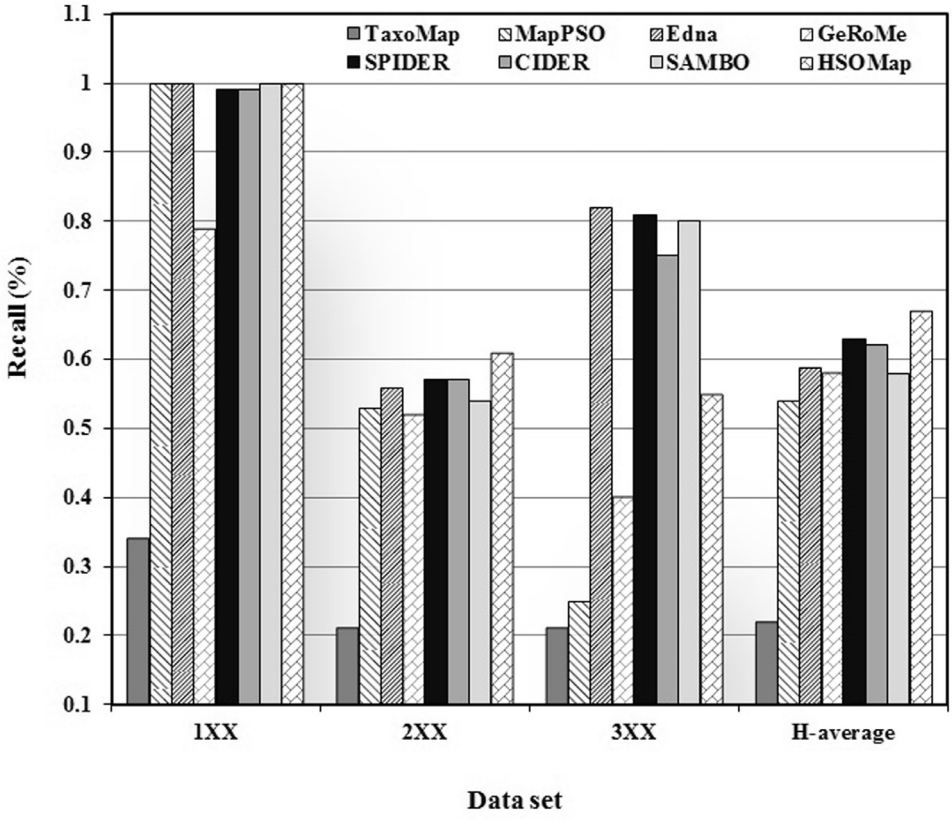

Fig. 10. The average recall of the HSOMap and the baseline methods participated in for the OAEI-08 campaign on 1XX, 2XX, and 3XX test cases.

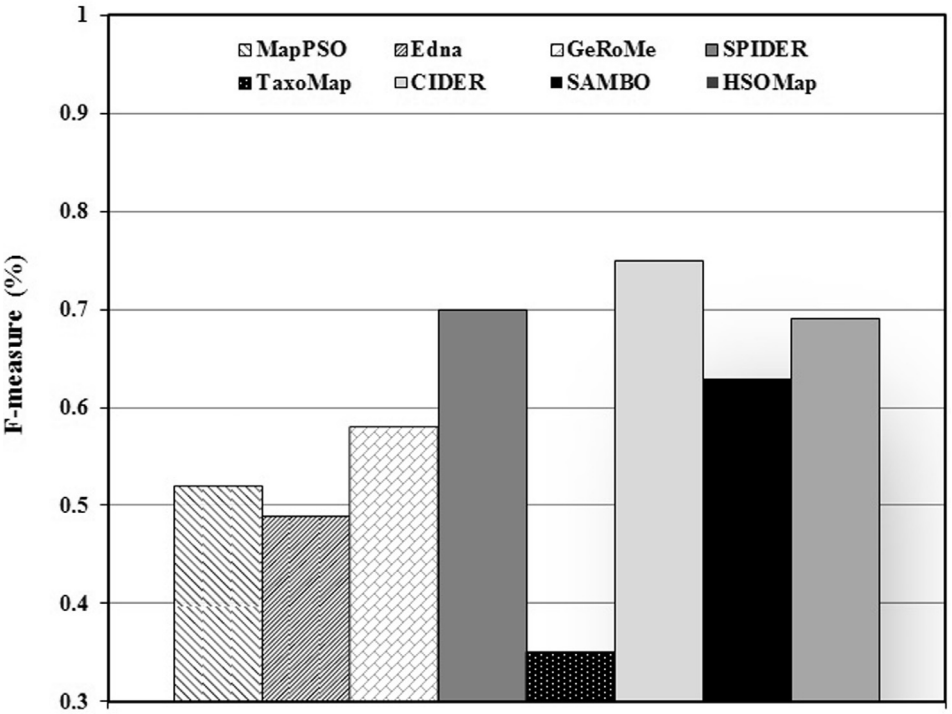

Fig. 11. The F-measure values of the HSOMap and the baseline methods.

The SPIDER has the same performance as the CIDER algorithm with the difference that it considers non-equivalence mappings. The tests of this set do not make sense for the Scarlet module of SPIDER as a comparison is sought between modified versions of the same ontology; therefore, it has the same results as the CIDER for this type of tests.

For most of the tests in this category, the performance of GeRoMe was satisfying, but for some tasks, especially those without any linguistic information, it produced disappointing results. This is due to the fact that the GeRoMe does not take into account the overall structure of the ontology. For example, in the case of #202, where no linguistic information at all was available, it produced very low results.

The SAMBO uses a general thesaurus, i.e., WordNet, to enhance the similarity measure by looking up the hypernym relationships of the pairs of words in WordNet and obtained good results in many cases. In this category of tests, the MapPSO performs worst for benchmark cases #201 and #202 in which all names and comments of ontologies are unavailable, as the main focus of the MapPSO is based on linguistic features, such as string distance and WordNet distance. The results of these test cases are not as positive, as the quality of the alignment decreases with the number of features that provide linguistic features to exploit.

Test 3XX. The evaluations with real ontologies also show the recall of the TaxoMap is low because it does not take into account properties and instances and only generates mappings between concepts. The real tests also show that the precision is high.

The results on real ontologies show the very good behavior of the CIDER in terms of precision (i.e., 0.9), while keeping an acceptable recall (i.e, 0.73).

The test set 3XX contains a few non-equivalence mappings in the reference alignment; therefore, it is a good candidate for evaluating the SPIDER system. The results of the SPIDER show that while recall increases for those cases where the reference alignments also contain subsumption relations when compared to the CIDER, precision is heavily affected.

For test cases #301 and #304, GeRoMe produced quite reasonable results by using the WordNet matcher for detecting synonyms. Test case #303 could not be processed by GeRoeE as there was a problem with importing this ontology into the generic representation of GeRoMe. The precision and recall values for the MapPSO algorithm vary between 0 and 0.6 in these types of test cases, and no uniform results can be derived.

8. Conclusions

We have studied the combinatorial ontology mapping problem and developed a general framework to effectively find a near-optimal matching between two input ontologies. The proposed HSOMap method is a symbiosis between the combinatorial ontology mapping techniques and the harmony search optimization algorithm. Roughly speaking, the combinatorial ontology mapping problem is modeled as a global optimization problem, and the harmony search algorithm is applied to extract an approximate optimal solution. The main ingredient of the method is to exploit different similarity measures to devise an effective fitness function to guide the optimization process and balance the exploration and exploitation in searching for the optimal solution. We have also modified the pitch adjusting process of the harmony search algorithm to adapt it to the discrete nature of the ontology mapping problem. We conducted a set of experiments to analyze and evaluate the performance of the HSOMap algorithm on benchmark ontologies. The experimental results revealed that the proposed algorithm has good performance and can be used in a single goal-driven way versus using composite mapping algorithms.

There are several directions in which this work could be extended. We also believe further investigation is needed into this problem. We plan to study a multi-objective strategy in order to avoid unwanted deviations from precision and recall values. Another very important research direction would be to design more scalable algorithms to be able to map large-scale ontologies. Also, it is worth investigating whether the idea of exploiting side information about ontological entities could be further improved to better take advantage of external sources of knowledge about ontologies. One direction would be to utilize machine learning techniques, and in particular, distance metric learning to learn a unified Mahalanobis distance to measure the similarity of ontological entities instead of aggregating different similarity measures in a linear fashion.

Acknowledgments

The authors thank anonymous reviewers and editor-in-chief for their constructive comments which led to the overall improvement of this paper. Also, the authors are indebted to JoAnn Woods for her literary contributions.

References

[1] I. Akbari, M. Fathian, A novel algorithm for ontology matching, J. Inf. Sci. 36 (3) (2010) 324–334.
[2] A. Alasoud, V. Haarslev, N. Shiri, An empirical comparison of ontology matching techniques, J. Inf. Sci. 35 (4) (2009) 379–397.
[3] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and the hardness of approximation problems, J. ACM (JACM) 45 (3) (1998) 501–555.
[4] J. Bock, J. Hettenhausen, Discrete particle swarm optimisation for ontology alignment, Inf. Sci. 192 (2012) 152–173.
[5] J. Bock, P. Liu, J. Hettenhausen, Mappso results for oaei 2008, in: Proceedings of the the 7th International Semantic Web Conference, 2008.
[6] P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, S. Tessaris, D2. 2.1 specification of a common framework for characterizing alignment, Knowl. Web Consor. 1 (2004) 1–36.
[7] P. Bouquet, L. Serafini, S. Zanobini, Peer-to-peer semantic coordination, Web Semant. Sci. Serv. Agents World Wide Web 2 (1) (2004) 81–97.

[8] C. Caraciolo, J. Euzenat, L. Hollink, R. Ichise, A. Isaac, V. Malaise, C. Meilicke, J. Pane, P. Shvaiko, H. Stuckenschmidt, et al., Results of the ontology alignment evaluation initiative 2008, in: Proceedings of the 3rd ISWC Workshop on Ontology Matching (OM), 2008, pp. 73–119.
[9] W.W. Cohen, P. Ravikumar, S.E. Fienberg, et al., A comparison of string distance metrics for name-matching tasks, in: Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), vol. 47, 2003, pp. 1–6.
[10] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, Syst. Man Cybern. Part B: Cybern. IEEE Trans. 41 (1) (2011) 89–106.
[11] Y. Ding, S. Foo, Ontology research and development. Part 1-a review of ontology generation, J. Inf. Sci. 28 (2) (2002) 123–136.
[12] H.-H. Do, E. Rahm, Coma: a system for flexible combination of schema matching approaches, in: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 610–621.
[13] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Halevy, Learning to match ontologies on the semantic web, Int. J. Very Large Data Bases 12 (4) (2003) 303–319.
[14] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Learning to map between ontologies on the semantic web, in: Proceedings of the 11th international conference on World Wide Web, ACM, 2002, pp. 662–673.
[15] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Ontology matching: A machine learning approach, Handbook on Ontologies in Information Systems (2004) 397–416.
[16] M. Ehrig, S. Staab, Qom–quick ontology mapping, in: The Semantic Web–ISWC 2004, Springer, 2004, pp. 683–697.
[17] J. Euzenat, P. Shvaiko, Ontology Matching, vol. 18, Springer, 2007.
[18] L. Feiyu, State of the Art : Automatic Ontology Matching, Technical Report 2007:1, School of Engineering, Jonkoping University, JTH, Computer and Electrical Engineering, 2007.
[19] R. Forsati, A. Haghighat, M. Mahdavi, Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing, Comput. Commun. 31 (10) (2008) 2505–2519.
[20] R. Forsati, M. Mahdavi, M. Shamsfard, M. Reza Meybodi, Efficient stochastic algorithms for document clustering, Inf. Sci. 220 (2013) 269–291.
[21] R. Forsati, M. Shamsfard, Novel harmony search-based algorithms for part-of-speech tagging, Knowl. Inf. Syst. (2014) 1–28.
[22] Z.W. Geem, Novel derivative of harmony search algorithm for discrete design variables, Appl. Math. Comput. 199 (1) (2008) 223–230.
[23] Z.W. Geem, Music-inspired harmony search algorithm, Stud. Comput. Intell., Vol. 191, Springer, 2009 http://link.springer.com/book/10.1007/978-3-642-00185-7.
[24] Z.W. Geem, C.-L. Tseng, Y. Park, Harmony search for generalized orienteering problem: best touring in china, in: Advances in Natural Computation, Springer, 2005, pp. 741–750.
[25] F. Giunchiglia, P. Shvaiko, M. Yatskevich, S-match: an algorithm and an implementation of semantic matching, in: The Semantic Web: Research and Applications, Springer, 2004, pp. 61–75.
[26] J. Gracia, J. Bernad, E. Mena, Ontology Matching with CIDER: Evaluation Report for OAEI (2008) Ontol. Match.
[27] T.R. Gruber, et al., Toward principles for the design of ontologies used for knowledge sharing, Int. J. Human Comput. Stud. 43 (5) (1995) 907–928.
[28] M. Hadwan, M. Ayob, N.R. Sabar, R. Qu, A harmony search algorithm for nurse rostering problems, Inf. Sci. 233 (2013) 126–140.
[29] S.B.R.C. Hamdi F, Zargayouna H, MapPSO results for OAEI 2008, in: TaxoMap in the OAEI 2008 Alignment Contest, 2008.
[30] W. Hu, N. Jian, Y. Qu, Y. Wang, Gmo: A graph matching for ontologies, in: Proceedings of K-CAP Workshop on Integrating Ontologies, 2005, pp. 41–48.
[31] J. Huang, J. Dang, J.M. Vidal, M.N. Huhns, Ontology matching using an artificial neural network to learn weights, in: Proceedings of the IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition, 2007.
[32] Y.R. Jean-Mary, E.P. Shironoshita, M.R. Kabuka, Ontology matching with semantic verification, Web Semant. Sci. Serv. Agents World Wide Web 7 (3) (2009) 235–251.
[33] D. Kensche, C. Quix, M.A. Chatti, M. Jarke, GeRoMe: a generic role based metamodel for model management, in: Journal on Data Semantics VIII, Vol. 4380, Springer, 2007, pp. 82–117.
[34] P. Lambrix, H. Tan, Sambo- a system for aligning and merging biomedical ontologies, Web Semant. Sci. Serv. Agents World Wide Web 4 (3) (2006) 196–206.
[35] J. Li, J. Tang, Y. Li, Q. Luo, Rimom: a dynamic multistrategy ontology alignment framework, Knowl. Data Eng. IEEE Trans. 21 (8) (2009) 1218–1232.
[36] J. Madhavan, P.A. Bernstein, A. Doan, A. Halevy, Corpus-based schema matching, in: Proceedings of the 21st International Conference on Data Engineering, 2005. ICDE 2005, IEEE, 2005, pp. 57–68.
[37] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, in: Proceedings of the International Conference on Very Large Data Bases, 2001, pp. 49–58.
[38] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[39] M. Mao, Y. Peng, M. Spring, An adaptive ontology mapping approach with neural network based constraint satisfaction, Web Semant. Sci. Serv. Agents World Wide Web 8 (1) (2010) 14–25.
[40] J. Martinez-Gil, E. Alba, J.F. Aldana-Montes, Optimizing ontology alignments by using genetic algorithms, in: Proceedings of the workshop on nature based reasoning for the semantic Web. Karlsruhe, Germany, 2008, pp. 1–14.
[41] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K.J. Miller, Introduction to wordnet: An on-line lexical database, Int. J. Lexicogr. 3 (4) (1990) 235–244.
[42] M.G. Omran, M. Mahdavi, Global-best harmony search, Appl. Math. Comput. 198 (2) (2008) 643–656.
[43] P. Pantel, D. Lin, Discovering word senses from text, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining, ACM, 2002, pp. 613–619.
[44] G. Pirro, A semantic similarity metric combining features and intrinsic information content, Data Knowl. Eng. 68 (11) (2009) 1289–1308.
[45] V. Qazvinian, H. Abolhassani, B.B. Hariri, et al., Evolutionary coincidence-based ontology mapping extraction, Expert Syst. 25 (3) (2008) 221–236.
[46] Y. Qu, W. Hu, G. Cheng, Constructing virtual documents for ontology matching, in: Proceedings of the 15th International Conference on World Wide Web, ACM, 2006, pp. 23–31.
[47] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB J. 10 (4) (2001) 334–350.
[48] D. Ritze, J. Volker, C. Meilicke, O. Svab-Zamazal, Linguistic analysis for complex ontology matching, Ontol. Matching 1 (2010) 1–12.
[49] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manag. 24 (5) (1988) 513–523.
[50] S.S. Shreem, S. Abdullah, M.Z.A. Nazri, Hybridising harmony search with a Markov blanket for gene selection problems, Inf. Sci. 258 (2014) 108–121.
[51] P. Shvaiko, J. Euzenat, A survey of schema-based matching approaches, in: Journal on Data Semantics IV, Vol. 3730, Springer, 2005, pp. 146–171.
[52] X.L. Sun, E. Rose, Automated schema matching techniques: an exploratory study, Res. Lett. Inf. Math. Sci. 4 (2003) 113–136.
[53] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, K. Wang, Using Bayesian decision for ontology mapping, Web Semant. Sci. Serv. Agents World Wide Web 4 (4) (2006) 243–262.
[54] R. Trillo, J. Gracia, M. Espinoza, E. Mena, Discovering the semantics of user keywords, J. UCS 13 (12) (2007) 1908–1935.
[55] J. Wang, Z. Ding, C. Jiang, Gaom: genetic algorithm based ontology matching, in: Proceedings of the IEEE Asia-Pacific Conference on Services Computing, IEEE, 2006, pp. 617–620.
[56] L. Wang, R. Yang, Y. Xu, Q. Niu, P.M. Pardalos, M. Fei, An improved adaptive binary harmony search algorithm, Inf. Sci. 232 (2013) 58–87.
[57] R. Zhang, L. Hanzo, Iterative multiuser detection and channel decoding for ds-cdma using harmony search, IEEE Lett. Signal Process. 16 (10) (2009) 917–920.