IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 6, 2020, accepted February 28, 2020, date of publication March 2, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977763

A Compact Brain Storm Algorithm for Matching Ontologies

XINGSI XUE[1] AND JIAWEI LU[2]
[1]Fujian Key Laboratory for Automotive Electronics and Electric Drive, Fujian University of Technology, Fuzhou 350118, China
[2]College of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China

Corresponding author: Xingsi Xue (jack8375@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61503082, in part by the Natural Science Foundation of Fujian Province under Grant 2016J05145, in part by the Program for New Century Excellent Talents in Fujian Province University under Grant GY-Z18155, in part by the Program for Outstanding Young Scientific Researcher in Fujian Province University under Grant GY-Z160149, and in part by the Scientific Research Foundation of Fujian University of Technology under Grant GY-Z17162 and Grant GY-Z15007.

ABSTRACT An ontology can formally present the domain knowledge by specifying the domain concepts and their relationships, which is a kernel technique for addressing the data heterogeneity issue in the semantic web. However, since existing ontologies are developed and maintained independently by different communities, a concept and its relationship with the others could be described in different ways, yielding the ontology heterogeneity problem. To solve this problem, in this work, we formally construct an optimal model for it, and propose a similarity measure for distinguishing identical ontology entities. Since determining the high-quality ontology alignment is a complex process, we propose to utilize a Brain Storm Optimization algorithm (BSO) to optimize the alignment. BSO is a recently developed Swarm Intelligence algorithm (SI), which can effectively solve the complex optimization problem by imitating the human's idea generating process. However, classic BSO needs to cluster various ideas in each generation and carry out the evolving operators on all ideas, which increases the computational complexity. To improve the efficiency of BSO-based ontology matcher, a Compact BSO (CBSO) is further proposed, which can reduce the memory consumption by utilizing the probabilistic representation on the idea cluster, and improve the algorithm's speed through the compact crossover operator and perturbation operator. The experiment uses the benchmark track provided by the Ontology Alignment Evaluation Initiative (OAEI) to test our approach's performance. The comparisons among the state-of-the-art ontology matchers and our proposal show that CBSO-based ontology matcher can efficiently determine high-quality ontology alignments.

INDEX TERMS Ontology matching, compact brain storm optimization algorithm, ontology alignment evaluation initiative.

I. INTRODUCTION

An ontology can formally present the domain knowledge by specifying the domain concepts and their relationships, which is a kernel technique for addressing the data heterogeneity issue in the semantic web [1]. Fig. 1 shows an example of two ontologies. In the figure, the rectangle represents the class, e.g. ''Electronics'', ''Personal Computers'' and ''Microprocessors'', the lines between two classes represents their relationship ''has a'', and each class has the data properties to describe its features, e.g. ''Microprocessors'' has ''PID'', ''Name'', ''Quantity'' and ''Price''. However, since the ontologies are developed and maintained independently by different communities, they might use the same term to define different class, e.g. ''Accessories'', or utilize different words to define the same class, e.g. ''Personal Computers'' and ''PC'', yielding the ontology heterogeneity problem. To support the inter-operation between ontology-based intelligent applications, it is critical to determine the relationships between two ontologies' entities, e.g. the equivalence relationship ($\equiv$) between ''Personal Computers'' and ''PC'', the subsumption relationship ($\geq$) between ''Photo and Cameras'' and ''Digital Cameras'', the disjointness relationship ($\perp$) between ''Microprocessors'' and ''PC board'',

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

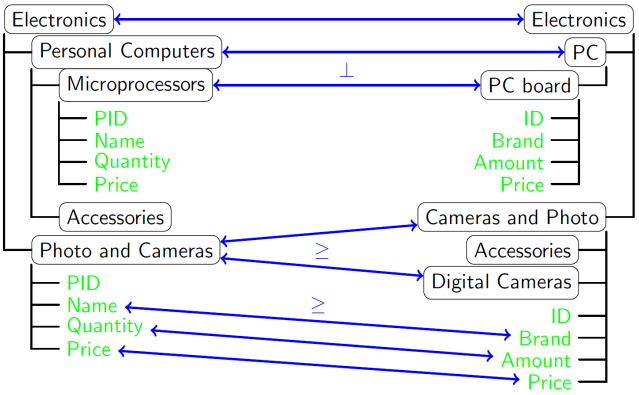


FIGURE 1. An example of matching two ontologies.

and further bridge the semantic gap between two ontologies. Ontology matching can determine the identical entities in an automatic or semi-automatic way, which is an effective technique for solving the ontology heterogenous problem [2].

Since the ontology could be described through its architecture graph (the nodes represent the concepts and instances, and the edges represent the relationships between them), the ontology matching problem could be seen as the determination of the largest isomorphic sub graph between two architecture graphs of two ontologies to be matched. Supposing $|O_1|$ and $|O_2|$ are respectively the cardinalities of two ontologies, the computational complexity of an ontology matcher usually is $O(|O_1|\cdot|O_2|)$, which would yield the out of memory error and long runtime when the scale of ontology entities is large. In addition, the terminologies used to define the ontology entities are semantically rich and ambiguous, e.g. the same terminology can be applied to define different entities, which would cause many local optimal solutions. Due to the complexity of the ontology matching process, Swarm Intelligent algorithm (SI) becomes an effective method for solving ontology matching problem [1]. Recently, various SI algorithms, such as Evolutionary Algorithm (EA) [3], Particle Swarm Optimization algorithm (PSO) [4] and Artificial Bee Colony algorithm (ABC) [5], have been utilized to match the ontologies and obtain acceptable results. Brain Storm Optimization (BSO) algorithm is a recently developed SI [6], which imitates the human's idea generating process to solve the complex optimization problem. Being inspired by the success of BSO in various domains [7]–[12], in this work, we propose to use it to address the ontology matching problem. Beside the ontology alignment's quality, the matching efficiency of a matcher is also very important [13]. Classic BSO executes the clustering operation in each generation and carry out the evolving operators on all solutions, which increases the computational complexity. To improve the efficiency of BSO, in this paper, we further propose a Compact BSO (CBSO), which utilizes the probability vector to approximate the idea cluster and execute BSO's evolving process. CBSO can significantly reduce BSO's memory consumption because it does not need to execute the clustering process, and decrease the runtime through the compact crossover operator and perturbation operator.Moreover, we also propose a novel similarity measure to distinguish the heterogeneous ontology entities, which integrates three kinds of similarity metrics to ensure the alignment's quality. In particular, the contributions made in this paper are as follows:

- an optimal model for ontology matching problem is constructed,
- a hybrid similarity measure is proposed to calculate the entities' similarity,
- a CBSO is proposed to efficiently match the ontologies.

The rest of the paper is organized as follows: Section II presents the related work; Section III defines the basic concepts about ontology, ontology matching problem and similarity measure; Section IV describes in details the CBSO-based ontology matching technique; Section V shows the experimental results; and finally, Section VI draws the conclusion.

II. RELATED WORK

A. BRAIN STORM ALGORITHM

BSO was proposed by Shi [6] in 2011, which is to solve the complex problem through imitating the brain storming sessions. Two features of BSO make it outstanding: (1) the clustering operator that partitions all the ideas generated into different groups, and (2) the creating operator that produces new ideas through the grouped ideas. The recent researches mainly focus on improving the classic BSO's performance in terms of the above two features. Zhan *et al.* [14] proposed a simple grouping strategy to reduce the clustering process's computational complexity, and a new creating operator to generate high-quality idea. Cao *et al.* [15] and Li *et al.* [16], [17] applied a randomly clustering approach, which evenly partition the current ideas and new ideas, and they also proposed the new creating operators to better trade off the algorithm's exploitation and exploration. Zhou *et al.* [18] not only utilized the random grouping strategy, but also used a dynamic step-size schedule based creating operator. Cheng *et al.* [19] presented a partially re-initializing strategy based creating operator, which can ensure the population diversity during the evolving process. Cao *et al.* [20] used Differential Evolution (DE) and the step-size schedule based creating operator to improve the new idea's solution. Chen *et al.* [21] took the group's structure information into account and further proposed an affinity propagation based clustering operator to dynamically adjust each idea group's scale. Li *et al.* [22] proposed a vector grouping learning scheme and use it to improve BSO's population diversity and search efficiency. Song *et al.* [23] proposed a simple BSO algorithm with a periodic quantum learning strategy which utilized a fitness value based clustering strategy to reduce the clustering process's computational burden and resist premature convergence, and a simplified idea updating strategy and a quantum-behaved individual updating with periodic learning strategy to enrich the diversity of newly generated ideas. More recently, Aldhafeeri and Rahmat-Samii [24]

proposed a binary version of Brain Storm Optimization algorithm (BBSO) to address the, a discrete optimization problem. BBSO utilized the Hamming distance based binary grouping operation, and two binary evolving operator to generate new idea.

In general, BSO is an effective algorithm but not efficient in terms of speed and memory consumption. To improve the efficiency of BSO, in this work, we propose a CBSO, which utilizes the probabilistic distribution to represent various idea groups and perform the idea creation. A run of CBSO can significantly improve BSO's efficiency in terms of speed and memory consumption, while at the same time ensure the quality of solution.

B. SWARM INTELLIGENCE ALGORITHM BASED ONTOLOGY MATCHING TECHNIQUE

Many Machine Learning (ML) techniques have been applied to match ontologies to determine high-quality alignment, such as Logistic Regression (LR) [25], Neural Network (NN) [26], Word Embedding (WE) [27], Graph Embedding (GE) [28], Support Vector Machine (SVM) [29], Clustering Algorithm [30], Decision Tree (DT) [31] and so forth. Researchers also try to improve the matching efficiency through the high performance computing techniques such as Parallel Computing (PC) [32] and Cloud Computing (CC) [33]. Due to the complexity of the ontology matching process, recently, SI-based technique has become an efficient approach for determining high-quality ontology alignment.

The first generation of SI-based matchers aimed at solving the ontology meta-matching problem, i.e. how to determine the optimal parameters to aggregate different matchers and optimize the quality of obtained ontology alignment. Genetics for Ontology ALignments (GOAL) [34] was the first SI-based ontology meta-matcher, which used Evolutionary Algorithm (EA) to optimize the aggregating weight set of different ontology matchers. Ginsca and Iftene [35] proposed to use EA to optimize the all the parameters in the whole meta-matching process, which included the aggregating weight set and a threshold for filtering the final alignment. Xue and Wang [1], [36] introduced a new metric to measure the ontology alignment's quality, which did not require the utilization of golden standard alignment, and formally defined ontology meta-matching problem. Their approach was able to match multiple ontology pairs at a time and overcame three drawbacks of the EA-based meta-matchers. More recently, He *et al.* [5] used Artificial Bee Colony (ABC) algorithm to address the ontology meta-matching problem, whose results were better than the EA-based matchers.

Since ontology meta-matching matchers need to maintain several similarity metrics during the matching process, which consumes huge memory, the second generation of SI-based matchers focus on directly determine the optimal ontology entity correspondence set. GAOM (Genetic Algorithm based Ontology Matching) [37] was the first SI-based ontology entity matcher, which regarded two ontologies as two discrete entity sets and employed EA to determine the optimal alignment. Similarity, MapPSO [4] used Particle Swarm Optimization (PSO) to determine the optimal entity correspondence set. In particular, PSO introduced a new quality measure on the ontology alignment, which depended on the statistical results on the alignment. Alves *et al.* [38] utilized a Memetic Algorithm (MA), which combined EA with a local search process, to execute the instance-based ontology matching process. They first matched the instances and then propagated the similarity values from the instance level to the corresponding concept level. Chu *et al.* [39] proposed a new similarity measure that modeled two ontologies in a vector space and used the cosine distance to calculate two entities' similarity, and on this basis, they used EA to optimize the alignment's quality. Our proposal belongs to the second generation of SI-based ontology entity matcher. Different from the population-based SI, we utilize the compact encoding mechanism on the population to save the memory consumption, and use the compact evolving operators to improve the algorithm's searching efficiency. To the best of our knowledge, this is the first time that CBSO is proposed and utilized for solving the ontology matching problem.

III. PRELIMINARIES

A. ONTOLOGY AND ONTOLOGY MATCHING PROBLEM

An ontology defines the domain concepts and the relationships, which can be defined as a 3-tuple (C, R, I), where C, R and O are respective the sets of concept, relationship (such as "is-a" and "has-a") and instance. In generally, concept, relationship and instance are called entities. To bridge the semantic gap between two heterogenous ontologies, we need to execute an ontology matching process to determine the identical entity pairs, and the entity correspondence set obtained is called the ontology alignment.

An ontology alignment's quality can be measured by recall, precision and f-measure [40], but these metrics require a golden standard alignment, which is not available especially when the scale of ontology is huge. Since the recall value of an alignment is proportional to its cardinality, and the precision value is proportional to the average similarity value of the entity mappings, we utilize the following metrics to approximately measure an alignment's quality:

$$r(A) = \frac{|A|}{max\{|O_1|, |O_2|\}} \tag{1}$$

$$p(A) = \frac{\sum simValue_i}{|A|} \tag{2}$$

$$f(A) = \frac{2 \times r(A) \times p(A)}{r(A) + p(A)} \tag{3}$$

where functions $r(A)$, $p(A)$ and $f(A)$ respectively approximately measure the alignment A's recall, precision and f-measure, $|O_1|$, $|O_2|$ and $|A|$ are respectively the cardinalities of two ontologies O_1, O_2 and A, $simValue_i$ is the similarity value of the ith pair of entities.

Next, we construct an optimal model for the ontology matching problem:

$$\begin{cases} max \quad f(A) \\ s.t. \quad A = (a_1, a_2, \cdots, a_{|O_1|})^T \\ \qquad a_i \in \{1, 2, \cdots, |O_2|\}, \quad i = 1, 2, \cdots, |C_1| \end{cases} \tag{4}$$

where $|O_1|$ and $|O_2|$ respectively represent the cardinalities of two ontologies O_1 and O_2, $a_i, i = 1, 2, \cdots, |O_1|$ is the ith entity mapping, i.e. ith entity in O_1 is mapped to a_ith entity in O_2.

B. SIMILARITY MEASURE

Similarity measure takes as input two entities and outputs a real number in [0,1] reflecting their similarity. In particular, 1 means two entities are identical, while 0 means they are completely different. There are mainly three kinds of similarity measure: string-based similarity measure, dictionary-based similarity measure and context-based similarity measure. Currently, since simply using a single similarity measure can not ensure the result's confidence, usually, it is necessary to aggregate several similarity measures to ensure the alignment's quality. The similarity measure used in this work combines three categories of similarity measures mentioned above, which calculates two entities' similarity value through context-based similarity measure, and measures each context element pair's similarity by string-based and dictionary-based similarity measures. To be specific, for each concept in the ontology, we utilize its context information to construct a profile for it, which consists of the information from its direct ascendant and descendants. On this basis, similarity value $sim(e_1, e_2)$ between two entities e_1 and e_2 is calculated according to Eq. 5, which is a context-based similarity measure:

$$\frac{\sum_{i=1}^{|p_1|} \max_{j=1\cdots|p^2|}(sim'(p_{1,i}, p_{2,j})) + \sum_{j=1}^{|p_2|} \max_{i=1\cdots|p_1|}(sim'(p_{2,j}, p_{1,i})}{2 \times min(|p_1|, |p_2|)} \tag{5}$$

where:

- p_1 and p_2 are the profiles of e_1 and e_2, respectively, and $|p_1|$ and $|p_2|$ are the cardinalities of p_1 and p_2, respectively,
- $p_{1,i}$ and $p_{2,j}$ are respectively the ith element of p_1 and jth element of p_2,
- $sim'()$ computes the similarity through string-based similarity measure and dictionary-based similarity measure.

Given two string s_1 and s_2, before calculating their similarity value $sim'(s_1, s_2)$, we need to execute a pre-process on them: (1) remove the punctuation and stop-words, (2) split the strings into words and convert them into lower-case, (3) lemmatizing and stemming the words. Then, their similarity value calculated by soft TF-IDF [41], which is a string-based similarity measure that shows better performance in terms of both precision and recall in the recent study [42]. In this work, we improve the original soft TF-IDF by considering words equal based on the Wordnet instead of Jaro Winkler metric [43]. The new soft TF-IDF integrates the Wordnet [44] based similarity measure, which is a dictionary-based similarity measure, to improve the result's precision. In particular, we utilize the Extended Java WordNet Library (extJWNL)[1] to exploit the Wordnet which an electronic lexical database that puts various senses of words into the synonym sets, and two words are regarded as similar if they are exactly matched or synonymous. Last, threshold of soft TF-IDF is set as 0.8 according to Cheatham *et al.* [42], and the threshold of context-based similarity measure is 0.85 through the experiment.

C. POPULATION-BASED BRAIN STORM OPTIMIZATION ALGORITHM

In the following, the pseudo-code of the population-based BSO is presented in Algorithm 1:

In the beginning, ideas (or solutions) are initialized randomly. After that, in every generation, each idea $idea_i$ is updated by following four steps: (1) similar ideas are clustered into a group, and the best idea in each cluster is selected as this group's cluster center, (2) a new idea $newIdea_i$ is generated by selecting a cluster center according to $P_{one-cluster}$ or a randomly selected idea from a selected cluster according to $P_{one-center}$, or executing the crossover on two probabilistically selected cluster centers or two randomly selected ideas from two selected clusters according to $P_{two-centers}$, (3) execute the perturbation operator on $newIdea_i$, (4) updating $idea_i$ through comparing it with $newIdea_i$ in terms of their fitness values. In particular, function $rand(0, 1)$ generates a random number in [0,1]. One of the drawbacks of the original BSO is brought by the basic K-means clustering algorithm, which is used to imitate the people's group discussion during each generation. K-means clustering algorithm needs several iterations to cluster ideas into several groups, which increases the algorithm's computational complexity. For more details on BSO, please see also [6].

IV. COMPACT BRAIN STORM OPTIMIZATION ALGORITHM

In this section, we will present in details the CBSO, where each idea group is described by a Probability Vector (PV) [36], and the clustering process is simplified as the process of updating PV. In the next, we first describe the compact encoding mechanism, the crossover operator and the perturbation operator, and then we give the pseudo-code of CBSO.

A. COMPACT ENCODING MECHANISM

Due to the complex decoding process of decimal encoding mechanism, the binary encoding mechanism becomes the mainstream approach for encoding and decoding solutions for the discrete optimization problem. Since the Gray code is an intuitive and popular binary encoding mechanism in the computer science domain, in this work, we utilize it to

[1] https://sourceforge.net/projects/extjwnl

Algorithm 1 Population-Based Brain Storm Optimization Algorithm

Input: the population size N, the number of idea clusters m, three probabilities $P_{one-cluster}$, $P_{one-center}$ and $P_{two-centers}$
Output: the best cluster center

```
initialize and evaluate n ideas;
generation = 1;
while generation < MaxGeneration do
  partition N ideas into m clusters;
  select a cluster center for each cluster;
  for each idea idea_i do
    if rand(0, 1) < P_one-cluster then
      randomly select a cluster cluster_j;
      if rand(0, 1) < P_one-center then
        newIdea_i = center_j;
      else
        randomly select an idea idea_j^p from cluster_j;
        newIdea_i = idea_j^p;
      end if
    else
      randomly select two clusters cluster_j and cluster_k;
      randomly select two ideas idea_j^p and idea_k^q;
      if rand(0, 1) < P_two-centers then
        newIdea_i = crossover(center_j, center_k);
      else
        newIdea_i = crossover(idea_j^p, idea_k^q);
      end if
    end if
    newIdea_i = perturb(newIdea_i);
    if fitness(newIdea_i) > fitness(idea_i) then
      idea_i = newIdea_i;
    end if
  end for
  generation = generation + 1;
end while
return the cluster center with the best fitness value;
```

encode each entity corresponding. An example of the encoding and decoding process is shown in the Fig. 2. In this work, we need to encode an ontology alignment, i.e. a set of concept mappings. Since an entity correspondence's kernel elements are two mapped concepts, we can simply make use of their indices in the ontologies to encode it. Here, we empirically choose the Gray code, which is a binary encoding mechanism, to encode an alignment. As can be seen from Fig. 2, the index means the source concept index and the corresponding bit values are the target concept index that is encoded through Gray code, e.g. the source concept "uterine gland" with index 8 is mapped to target concept "Uterine_Gland" with index 6 whose Gray code is 110. In particular, Gray code 000 means a source concept is not mapped to any target concept.

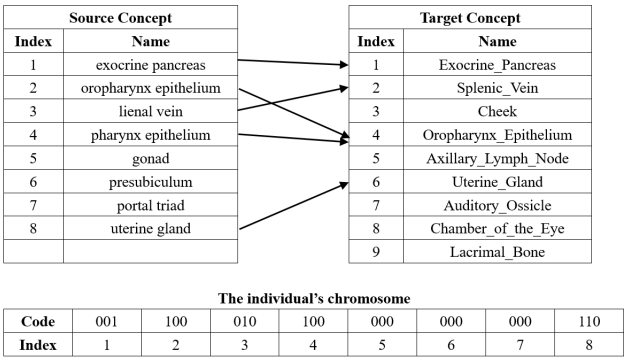


FIGURE 2. An example of encoding mechanism.

CBSO uses a PV to characterize each idea cluster. A PV's dimension is equal to the scale of an idea's gene bit, and each dimension's range is [0,1], In particular, the value in each PV's dimension represents a probability of being 1 on an idea's corresponding gene bit. Therefore, we can utilize a PV to generate different ideas in a group, which could be highly similar with each other. For example, given $PV = (0.2, 0.4, 0.6)^T$, we first generate three random real numbers in [0,1], say 0.5, 0.3 and 0.1. Since $0.5 > 0.2$ and $0.3 < 0.4$ and $0.1 < 0.6$, the newly generated idea is 011. If the new generated idea is selected as the cluster center, we will update PV by moving it to the cluster center. In particular, given an update step st, if the gene value of the cluster center is 1, the corresponding dimension number of PV will increase by st, otherwise decrease by st. In this example, assuming $st = 0.1$, the updated PV is $(0.1, 0.5, 0.7)^T$.

B. CROSSOVER OPERATOR

Given two parent ideas $idea_i$ and $idea_j$, the crossover operator generates the new idea on the basis of their distance. Since the ontology matching problem is a discrete optimization problem, the distance between $idea_i$ and $idea_j$ can be measured according to Eq. 6:

$$distance(idea_i, idea_j) = \sum_{k=1}^{|idea_i|} |idea_{i,k} - idea_{j,k}| \qquad (6)$$

where $|idea_i|$ is the cardinality of $idea_i$, $idea_{i,k}$ and $idea_{j,k}$ are respectively the kth gene bit value of $idea_i$ and $idea_j$. Next, the new idea $newIdea_i$ are generated by partly flipping the elements of $idea_i$, whose pseudo-code is given in Algorithm 2.

C. PERTURBATION OPERATOR

BSO's perturbation operator can be approximated through searching the vicinity range of $idea_i$. To this end, a perturbation matrix $M_{C\times D}$ is first constructed to generate neighbourhood of the $idea_i$, where $C = 5$ is the scale of neighbour population and D is the number of dimensions. For the sake of clarity, given a perturbation probability p_p, the pseudo-code of generating M is shown in Algorithm 3 [45].

Algorithm 2 Crossover Operator

$newIdea_i = idea_i$
for $k = 0; k < idea_i.length; k{+}{+}$ **do**
  if $idea_{i,k} != idea_{j,k}$ **then**
    append k to a index list $index$;
  end if
end for
$totalNum = round(rand(0, 1) \times distance(idea_i, idea_j))$;
$num = 0$;
$n = 0$;
while $num < totalNum$ **do**
  if $rand(0, 1) < P_{crossover}$ **then**
    $newIdea_{i,index[n]} = (newIdea_{i,index[n]} + 1)\ mod\ 2$;
    $index$.remove($index[n]$);
    $num = num + 1$;
  end if
  $n = (n + 1)\ mod\ index.length()$;
end while

Algorithm 3 Perturbation Matrix Construction

** initialize M **
for int $i = 0; i < C; i{+}{+}$ **do**
  for int $j = 0; j < D; j{+}{+}$ **do**
    $M_{ij} = 0$;
  end for
end for
** perturb M **
for int $i = 0; i < C; i{+}{+}$ **do**
  generate $j = round(rand(0, D))$;
  while $rand(0, 1) < p_p$ **do**
    $M_{ij} = 1$;
    $j = j + 1$;
    if $j == D$ **then**
      $j = 0$;
    end if
  end while
end for

Then, $\overline{M}$ is constructed by flipping the value in M, and the neighborhood of $idea_i$ can be generated according to Eq. 7.

$$\overrightarrow{idea_{neighbor}} = M \otimes \overrightarrow{idea_{elite}} + \overline{M} \otimes \overrightarrow{X} \tag{7}$$

where

$$\overrightarrow{idea_{basic}} = \begin{bmatrix} idea_i \\ idea_i \\ \cdots \\ idea_i \end{bmatrix}_{C \times D}, \quad \overrightarrow{X} = \begin{bmatrix} idea_1 \\ idea_2 \\ \cdots \\ idea_C \end{bmatrix}_{C \times D}$$

and $idea_j$, $j = 1, 2, \cdots, C$, is generated by PV^i, and the operator $\otimes$ is the multiplication of corresponding matrix elements.

Finally, we select the best idea from $idea_i$'s neighborhood as the new idea.

TABLE 1. A brief description on the benchmark track.

ID	Brief Description
101-104	Two same ontologies
201-210	Two ontologies with different lexical and linguistic features
221-247	Two ontologies with different structure feature
248-266	Two ontologies with different lexical, linguistic and structure features
301-304	Two real world ontologies

D. THE PSEUDO-CODE OF COMPACT BRAIN STORM OPTIMIZATION ALGORITHM

In the following, the pseudo-code of CBSO is given in Algorithm 4:

V. EXPERIMENT

A. EXPERIMENTAL SETUP

In the experiment, the benchmark track provided by the Ontology Alignment Evaluation Initiative (OAEI)[2] are used to test our approach's performance. Each testing case in the benchmark track consists of two ontologies to be matched and a golden standard alignment for evaluating the alignment's quality. Table 1 shows a brief description about the benchmark track.

We compare CBSO-based matcher with OAEI's participants and three state-of-the-art SI-based matchers, i.e. EA-based matcher [37], PSO-based matcher [4], ABC-based matcher [5] and BBSO-based matcher [24]. In order to compare with OAEI's participants, recall, precision and f-measure [40] are used to measure the alignment's quality. The parameter used by CBSO (see also Section IV-D) represent a trade-off setting obtained in an empirical way to ensure the highest average alignment quality in all testing cases, and the parameters of EA, PSO, ABC and BBSO are referred to their own literatures. In particular, EA, PSO, ABC, BBSO and CBSO's results shown in the tables are the mean values of thirty independent runs.

B. COMPARISON ON ALIGNMENT'S QUALITY

Tables 2 and 3 compares CBSO with SI-based matchers by carrying out the T-test statistical analysis [46] on f-measure, and Figure 3 compares CBSO with OAEI's participants and SI-based matchers in terms of recall, precision and f-measure.

Since all the algorithms are run for 30 independent executions on each testing case, the analysis has to consider the critical value $t_{0.025}$ for 29 degrees of freedom that is equal to 2.045. Therefore, as can be seen from Tables 2 and 3, the alignments obtained by CBSO are better than other SI-based matchers on all testing cases. In particular, CBSO's average standard deviation is lower than other SI-based matchers, which means it is more stable when optimizing the ontology alignments. Comparing with state-of-the-art SI-based approaches, CBSO works based on the probabilistic modeling of promising solutions, which makes it easier to predict the movements of the populations in the search

[2]http://oaei.ontologymatching.org/2016

TABLE 2. Comparison of the alignments in terms of f-measure and standard deviation *stDev*.

ID	f-measure ($stDev$) EA	f-measure ($stDev$) PSO	f-measure ($stDev$) ABC	f-measure ($stDev$) BBSO	f-measure ($stDev$) CBSO
101	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
103	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
104	0.99 (0.01)	0.98 (0.02)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
201	0.94 (0.01)	0.94 (0.01)	0.94 (0.01)	0.94 (0.00)	0.94 (0.01)
203	0.90 (0.01)	0.99 (0.01)	0.99 (0.01)	1.00 (0.00)	1.00 (0.00)
204	0.98 (0.01)	0.98 (0.01)	0.98 (0.02)	0.98 (0.00)	0.98 (0.01)
205	0.89 (0.04)	0.92 (0.06)	0.94 (0.02)	0.94 (0.02)	0.96 (0.01)
206	0.70 (0.03)	0.70 (0.01)	0.70 (0.01)	0.70 (0.00)	0.70 (0.00)
221	1.00 (0.01)	1.00 (0.01)	1.00 (0.01)	1.00 (0.00)	1.00 (0.00)
222	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
223	0.90 (0.01)	0.92 (0.01)	0.92 (0.02)	0.92 (0.03)	0.95 (0.02)
224	1.00 (0.01)	0.99 (0.01)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
225	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
230	0.90 (0.02)	0.95 (0.02)	0.90 (0.02)	0.98 (0.02)	0.98 (0.02)
231	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
301	0.75 (0.03)	0.70 (0.04)	0.70 (0.05)	0.73 (0.02)	0.77 (0.01)
302	0.71 (0.03)	0.65 (0.03)	0.72 (0.02)	0.74 (0.01)	0.74 (0.01)

TABLE 3. T-Test statistical analysis on the f-measure.

ID	$t-value$ ($p-value$) (CBSO, EA)	$t-value$ ($p-value$) (CBSO, PSO)	$t-value$ ($p-value$) (CBSO, ABC)	$t-value$ ($p-value$) (CBSO, BBSO)
101	0.00	0.00	0.00	0.00
103	0.00	0.00	0.00	0.00
104	5.47	4.55	0.00	0.00
201	0.00	0.00	0.00	0.00
203	54.77	5.47	5.47	0.00
204	0.00	0.00	0.00	0.00
205	9.29	3.60	4.89	4.89
206	0.00	0.00	0.00	0.00
221	0.00	0.00	0.00	0.00
222	0.00	0.00	0.00	0.00
223	12.24	7.34	5.80	0.00
224	0.00	5.47	0.00	0.00
225	0.00	0.00	0.00	0.00
230	15.49	5.80	15.49	0.00
231	0.00	0.00	0.00	0.00
301	3.46	9.29	7.51	9.79
302	4.55	13.67	2.14	0.00

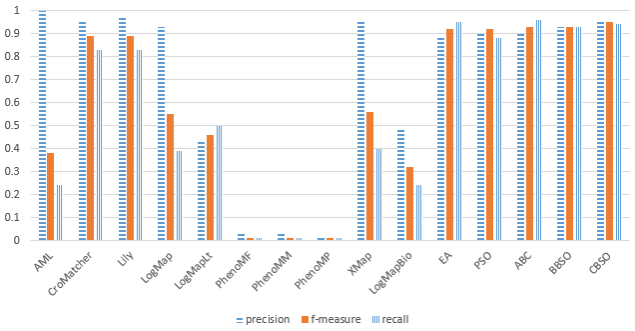


FIGURE 3. Comparison on the alignment's quality.

space and learn through the PVs to propose the new generation of ideas accordingly. Moreover, with the introduction of crossover and perturbation operators, CBSO is capable of effectively trading off the exploitation and exploration and learning more complex probabilistic model. Therefore, CBSO outperforms other SI-based approaches in terms of alignment quality.

As can be seen from Fig. 3, CBSO's f-measure is the highest among all OAEI's participants and SI-based matchers, which shows that CBSO can better trade off the recall and precision. In particular, the quality of alignment of CBSO is better than BBSO, which shows that CBSO's compact encoding mechanism and comapct operators can better trade off the algorithm's exploration and exploitation. Since none of the similarity measures can effectively distinguish all the heterogeneous concepts in any situations, it is necessary to aggregate several similarity measures to improve the result's precision. We utilize a hybrid similarity measure which combines three kinds of similarity measures to calculate the entity similarity value, and therefore CBSO's precision values are significantly higher than other matchers that only take into consideration one or two categories of similarity measure, such as CroMatcher, LogMap family, Pheno family, Lily and XMap. However, AML applies too many similarity measures that lead to the conflicting results, which decreases the recall value. Thus, how many matchers should be selected and combined to ensure the quality of the alignment is one of

Algorithm 4 Compact Brain Storm Optimization Algorithm

Input: the idea cluster's number $m = 5$, three probabilities $P_{one-cluster} = 0.6$, $P_{one-center} = 0.4$ and $P_{two-centers} = 0.5$, the length of an idea (or PV) *length*, the maximum generations $MaxGeneration = 2000$, the step length for updating PV $st = 0.1$
Output: the best cluster center

```
**** Initialization ****
for i = 0; i < m; i++ do
  for j = 0; j < length; j++ do
    PV_j^i = 0.5;
  end for
  generate the cluster center cs_i through PV^i;
end for
generation = 1;
**** Evolving Process ****
while generation < MaxGeneration do
  for i = 0; i < m; i++ do
    if rand(0, 1) < P_one-cluster then
      randomly select a PV PV^j;
      if rand(0, 1) < P_one-center then
        newIdea_i = cs_j;
      else
        generate an idea idea_j through PV^j;
        newIdea_i = idea_j;
      end if
    else
      **** Crossover ****
      randomly select two PVs PV^j and PV^k;
      generate two idea idea_j and idea_k through PV^j and
      PV^k, respectively;
      if rand(0, 1) < P_two-centers then
        newIdea_i = crossover(cs_j, cs_k);
      else
        newIdea_i = crossover(idea_j, idea_k);
      end if
    end if
    **** Perturbation ****
    newIdea_i = perturb(newIdea_i);
    **** Update Cluster Center ****
    [winner, loser] = compete(newIdea_i, cs_i);
    if winner == newIdea_i then
      cs_i = newIdea_i;
    end if
    **** Update PV ****
    for u = 0; u < length; u++ do
      if winner[u] == 1 then
        PV_u^i = PV_u^i + st;
      else
        PV_u^i = PV_u^i - st;
      end if
    end for
  end for
  generation = generation + 1;
end while
return the cluster center with the best fitness value;
```

TABLE 4. Comparison on the speed.

Testing Case	Runtime (second)	f-measure per second
AML	120	0.0031
CroMatcher	1100	0.0008
Lily	2211	0.0004
LogMap	194	0.0028
LogMapLt	96	0.0048
PhenoMF	1632	0.0000
PhenoMM	1743	0.0000
PhenoMP	1833	0.0000
XMap	123	0.0045
LogMapBio	54439	0.0000
EA	284	0.0032
PSO	271	0.0033
ABC	432	0.0021
BBSO	638	0.0014
CBSO	170	0.0055

our future work. In the next section, we will further compare CBSO with OAEI's participants and SI-based matchers in terms of speed and memory consumption.

C. COMPARISON ON SPEED AND MEMORY CONSUMPTION

In this section, we compare our approach with OAEI's participants and SI-based matchers in terms of speed and average memory consumption. Since OAEI official website provides no information on its participants' memory consumption, in Figure 4, we compare CBSO with SI-based matchers. In Table 4, a matcher's f-measure per second is calculated by dividing its mean f-measure by the mean runtime, which is a metric taken by OAEI to measure the matcher's efficiency.

Supposing $|O_1|$ and $|O_2|$ are respectively the cardinalities of two ontologies to be aligned, m is the cluster number, and N is the population size, the computational complexity of the SI-based matchers are as follows: (1) EA-based matcher: $O(|O_1| \cdot (log_2|O_2| + 1) \cdot N)$; (2) PSO-based matcher: $O(\min\{|O_1|, |O_2|\} \cdot (log_2 \min\{|O_1|, |O_2|\}) + 1) \cdot N)$; (3) ABC-based matcher: $O(|O_1| \cdot |O_2| \cdot N)$; (4) BBSO-based matcher: $O(|O_1| \cdot (log_2|O_2| + 1) \cdot m \cdot N)$; and (5) CBSO-based matcher: $O(|O_1| \cdot (log_2|O_2| + 1) \cdot m)$, where $log_2|O_2|$ is the length of Gray code to encode $|O_2|$ target entities. Thus, EA-based approach and PSO-based approach's performances are very closed to each other, ABC-based approach's calculation complexity is higher than others because it needs to maintain several similarity matrices, BBSO-based approach's computational complexity is relatively high since it have to execute the clustering operation beforehand, and CBSO-based approach utilize the PV to approximate the clustering process and execute the evolving process, whose computational complexity is the lowest. As can been seen from Table 4 and Fig. 4, due to the efficiency brought by the compact encoding mechanism and compact operators, CBSO's f-measure per second and the memory consumption is significantly lower than other SI-based matchers. The gains in the efficiency are achieved respectively due to CBSO's particular competitive learning, which is effective to lead the algorithm to determine the optimal solution, and the simplicity, which does not require all the mechanisms of a BSO,

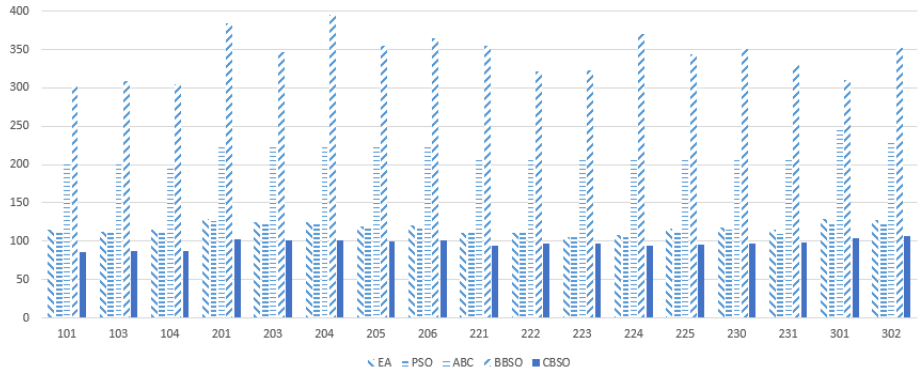


FIGURE 4. Comparison on the memory consumption.

rather the few steps in the algorithm are small and simple. To sum up, CBSO can efficiently determine high-quality ontology alignments.

VI. CONCLUSION

To efficiently match the ontologies, in this paper, we propose a CBSO-based ontology matching technique. CBSO reduces the memory consumption by utilizing the probabilistic representation of the idea cluster, and improves the speed by using the compact crossover operator and perturbation operator. The experiment uses the OAEI's benchmark track to test the performance of CBSO-based ontology matcher, and the experimental results show that CBSO-based ontology matcher is both effective and efficient.

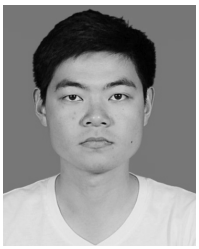
In the future, we will further study the technique that can adaptively select and combine various similarity measures according to different heterogeneity situation. Moreover, we will improve CBSO-based approach to match the large-scale ontologies such as biomedical ontolgies, which is an open challenge in the ontology matching domain. Another challenge in ontology matching domain is the problem of Instance Coreference Resolution (ICR), which requires matching large-scale instances in the Linked Open Data cloud (LOD). Currently, there is no SI-based technique that could effectively solve ICR, and we are also interested in addressing this challenge with CBSO.

REFERENCES

[1] X. Xue and Y. Wang, "Optimizing ontology alignments through a memetic algorithm using both MatchFmeasure and unanimous improvement ratio," *Artif. Intell.*, vol. 223, pp. 65–81, Jun. 2015.
[2] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, and A. Gómez-Rodríguez, "Ontology matching: A literature review," *Expert Syst. Appl.*, vol. 42, no. 2, pp. 949–971, Feb. 2015.
[3] G. Acampora, V. Loia, and A. Vitiello, "Enhancing ontology alignment through a memetic aggregation of similarity measures," *Inf. Sci.*, vol. 250, pp. 1–20, Nov. 2013.
[4] J. Bock and J. Hettenhausen, "Discrete particle swarm optimisation for ontology alignment," *Inf. Sci.*, vol. 192, pp. 152–173, Jun. 2012.
[5] Y. He, X. Xue, and S. Zhang, "Using artificial bee colony algorithm for optimizing ontology alignment," *J. Inf. Hiding Multimedia Signal Process.*, vol. 8, no. 4, pp. 766–773, 2017.
[6] Y. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, 2011, pp. 303–309.
[7] K. Lenin, B. R. Reddy, and M. S. Kalavathi, "Brain storm optimization algorithm for solving optimal reactive power dispatch problem," *Int. J. Res. Electron. Commun. Technol.*, vol. 1, no. 3, pp. 25–30, 2014.
[8] H. Qiu, H. Duan, and Y. Shi, "A decoupling receding horizon search approach to agent routing and optical sensor tasking based on brain storm optimization," *Optik*, vol. 126, nos. 7–8, pp. 690–696, Apr. 2015.
[9] K. Madheswari, N. Venkateswaran, and V. Sowmiya, "Visible and thermal image fusion using curvelet transform and brain storm optimization," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2016, pp. 2826–2829.
[10] Y. Wu, X. Wang, Y. Fu, and Y. Xu, "Difference brain storm optimization for combined heat and power economic dispatch," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2017, pp. 519–527.
[11] E. Dolicanin, I. Fetahovic, E. Tuba, R. CAPOR-HROSIK, and M. Tuba, "Unmanned combat aerial vehicle path planning by brain storm optimization algorithm," *Stud. Informat. Control*, vol. 27, no. 1, pp. 15–24, Mar. 2018.
[12] F. Pourpanah, Y. Shi, C. P. Lim, Q. Hao, and C. J. Tan, "Feature selection based on brain storm optimization for data classification," *Appl. Soft Comput.*, vol. 80, pp. 761–775, Jul. 2019.
[13] X. Xue and J. Chen, "Optimizing ontology alignment through hybrid population-based incremental learning algorithm," *Memetic Comput.*, vol. 11, no. 2, pp. 209–217, Mar. 2019.
[14] Z.-H. Zhan, J. Zhang, Y.-H. Shi, and H.-l. Liu, "A modified brain storm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.
[15] Z. Cao, Y. Shi, X. Rong, B. Liu, Z. Du, and B. Yang, "Random grouping brain storm optimization algorithm with a new dynamically changing step size," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2015, pp. 357–364.
[16] C. Li, Z. Luo, Z. Song, F. Yang, J. Fan, and P. X. Liu, "An enhanced brain storm sine cosine algorithm for global optimization problems," *IEEE Access*, vol. 7, pp. 28211–28229, 2019.
[17] C. Li, Z. Song, J. Fan, Q. Cheng, and P. X. Liu, "A brain storm optimization with multi-information interactions for global optimization problems," *IEEE Access*, vol. 6, pp. 19304–19323, 2018.
[18] D. Zhou, Y. Shi, and S. Cheng, "Brain storm optimization algorithm with modified step-size and individual generation," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, 2012, pp. 243–252.
[19] S. Cheng, Y. Shi, Q. Qin, Q. Zhang, and R. Bai, "Population diversity maintenance in brain storm optimization algorithm," *J. Artif. Intell. Soft Comput. Res.*, vol. 4, no. 2, pp. 83–97, Apr. 2014.
[20] Z. Cao, X. Hei, L. Wang, Y. Shi, and X. Rong, "An improved brain storm optimization with differential evolution strategy for applications of ANNs," *Math. Problems Eng.*, vol. 2015, Sep. 2015, Art. no. 923698.
[21] J. Chen, S. Cheng, Y. Chen, Y. Xie, and Y. Shi, "Enhanced brain storm optimization algorithm for wireless sensor networks deployment," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2015, pp. 373–381.
[22] C. Li, D. Hu, Z. Song, F. Yang, Z. Luo, J. Fan, and P. X. Liu, "A vector grouping learning brain storm optimization algorithm for global optimization problems," *IEEE Access*, vol. 6, pp. 78193–78213, 2018.
[23] Z. Song, J. Peng, C. Li, and P. X. Liu, "A simple brain storm optimization algorithm with a periodic quantum learning strategy," *IEEE Access*, vol. 6, pp. 19968–19983, 2018.

[24] A. Aldhafeeri and Y. Rahmat-Samii, "Brain storm optimization for electromagnetic applications: Continuous and discrete," *IEEE Trans. Antennas Propag.*, vol. 67, no. 4, pp. 2710–2722, Apr. 2019.
[25] N. Alboukaey and A. Joukhadar, "Ontology matching as regression problem," *J. Digit. Inf. Manage.*, vol. 16, no. 1, pp. 33–42, 2018.
[26] M. A. Khoudja, M. Fareh, and H. Bouarfa, "Ontology matching using neural networks: Survey and analysis," in *Proc. Int. Conf. Appl. Smart Syst. (ICASS)*, Nov. 2018, pp. 1–6.
[27] M. T. Dhouib, C. F. Zucker, and A. G. Tettamanzi, "An ontology alignment approach combining word embedding and the radius measure," in *Proc. Int. Conf. Semantic Syst.* Cham, Switzerland: Springer, 2019, pp. 191–197.
[28] A. Assi, H. Mcheick, A. Karawash, and W. Dhifli, "Context-aware instance matching through graph embedding in lexical semantic space," *Knowl.-Based Syst.*, vol. 186, Dec. 2019, Art. no. 104925.
[29] F. Ali, K.-S. Kwak, and Y.-G. Kim, "Opinion mining based on fuzzy domain ontology and support vector machine: A proposal to automate online review classification," *Appl. Soft Comput.*, vol. 47, pp. 235–250, Oct. 2016.
[30] X. Xue and J.-S. Pan, "A segment-based approach for large-scale ontology matching," *Knowl. Inf. Syst.*, vol. 52, no. 2, pp. 467–484, Jan. 2017.
[31] S. Amrouch, S. Mostefai, and M. Fahad, "Decision trees in automatic ontology matching," *Int. J. Metadata, Semantics Ontologies*, vol. 11, no. 3, pp. 180–190, 2016.
[32] T. B. Araújo, C. E. S. Pires, T. P. da Nóbrega, and D. C. Nascimento, "A fine-grained load balancing technique for improving partition-parallel-based ontology matching approaches," *Knowl.-Based Syst.*, vol. 111, pp. 17–26, Nov. 2016.
[33] M. B. Amin, W. A. Khan, S. Hussain, D.-M. Bui, O. Banos, B. H. Kang, and S. Lee, "Evaluating large-scale biomedical ontology matching over parallel platforms," *IETE Tech. Rev.*, vol. 33, no. 4, pp. 415–427, 2016.
[34] J. Martinez-Gil and J. F. Aldana-Montes, "Evaluation of two heuristic approaches to solve the ontology meta-matching problem," *Knowl. Inf. Syst.*, vol. 26, no. 2, pp. 225–247, 2011.
[35] A.-L. Ginsca and A. Iftene, "Using a genetic algorithm for optimizing the similarity aggregation step in the process of ontology alignment," in *Proc. 9th Roedunet Int. Conf.*, Sibiu, Romania, 2010, pp. 118–122.
[36] X. Xue and J.-S. Pan, "A compact co-evolutionary algorithm for sensor ontology meta-matching," *Knowl. Inf. Syst.*, vol. 56, no. 2, pp. 335–353, 2018.
[37] J. Wang, Z. Ding, and C. Jiang, "GAOM: Genetic algorithm based ontology matching," in *Proc. IEEE Asia–Pacific Conf. Services Comput. (APSCC)*, Guangzhou, China, Dec. 2006, pp. 617–620.
[38] A. Alves, K. Revoredo, and F. Bai ao, "Ontology alignment based on instances using hybrid genetic algorithm," in *Proc. 7th Int. Conf. Ontology Matching*, vol. 946, 2012, pp. 242–243.
[39] S.-C. Chu, X. Xue, J.-S. Pan, and X. Wu, "Optimizing ontology alignment in vector space," *J. Internet Technol.*, vol. 21, pp. 15–22, Jan. 2020.
[40] C. J. V. Rijsberge, *Information Retrieval*. London, U.K.: Univ. Glasgow, 1975.
[41] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *IEEE Intell. Syst.*, vol. 18, no. 5, pp. 16–23, Sep. 2003.
[42] M. Cheatham and P. Hitzler, "String similarity metrics for ontology alignment," in *Proc. Int. Semantic Web Conf.* Berlin, Germany: Springer, 2013, pp. 294–309.
[43] W. E. Winkler, "Matching and record linkage," in *Business Survey Methods*, vol. 1. New York, NY, USA: Wiley, 1995, pp. 355–384.
[44] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[45] J.-S. Pan, Z. Meng, S.-C. Chu, and H.-R. Xu, "Monkey king evolution: An enhanced ebb-tide-fish algorithm for global optimization and its application in vehicle navigation under wireless sensor network environment," *Telecommun. Syst.*, vol. 65, no. 3, pp. 351–364, 2017.
[46] J. S. Hampton, *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*. New York, NY, USA: McGraw-Hill, 1990.

XINGSI XUE received the B.S. degree in software engineering from Fuzhou University, China, in 2004, the M.S. degree in computer application technology from the Renmin University of China, China, in 2009, and the Ph.D. degree in computer application technology from Xidian University, China, in 2014. He is currently a Professor with the College of Information Science and Engineering, Fujian University of Technology. He is also a Kernel Member of the Intelligent Information Processing Research Center, Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian Key Laboratory for Automotive Electronics and Electric Drive, Fujian University of Technology. His research interests include intelligent computation, data mining, and large-scale ontology matching technology. He is a member of ACM. He received the 2017 ACM Xi'an Rising Star Award and the IIH-MSP 2016 Excellent Paper Award.



JIAWEI LU received the bachelor's degree from the Wuxi Electrical and Mechanical College, Jiangsu University, China, in 2018. He is currently a Graduate Student at the College of Information and Science Engineering, Fujian University of Technology. His research domains include intelligent computation, nature language processing, and ontology matching technique.