Knowledge and Information Systems (2021) 63:2919–2946
https://doi.org/10.1007/s10115-021-01613-0

REGULAR PAPER

An experimental analysis on evolutionary ontology meta-matching

Nicolas Ferranti[1,2] · **Jairo Francisco de Souza**[1] · **Stênio Sã Rosário Furtado Soares**[3]

Received: 1 October 2020 / Revised: 16 September 2021 / Accepted: 19 September 2021 /
Published online: 23 October 2021

Abstract
Every year, new ontology matching approaches have been published to address the heterogeneity problem in ontologies. It is well known that no one is able to stand out from others in all aspects. An ontology meta-matcher combines different alignment techniques to explore various aspects of heterogeneity to avoid the alignment performance being restricted to some ontology characteristics. The meta-matching process consists of several stages of execution, and sometimes the contribution/cost of each algorithm is not clear when evaluating an approach. This article presents the evaluation of solutions commonly used in the literature in order to provide more knowledge about the ontology meta-matching problem. Results showed that the more characteristics of the entities that can be captured by similarity measures set, the greater the accuracy of the model. It was also possible to observe the good performance and accuracy of local search-based meta-heuristics when compared to global optimization meta-heuristics. Experiments with different objective functions have shown that semi-supervised methods can shorten the execution time of the experiment but, on the other hand, bring more instability to the result.

Keywords Ontology meta-matching · Metaheuristic-based ontology matching · Evolutionary ontology matching · Semantic web

✉ Nicolas Ferranti
nicolas1@ice.ufjf.br; nicolas.ferranti@wu.ac.at

Jairo Francisco de Souza
jairo.souza@ice.ufjf.br

Stênio Sã Rosário Furtado Soares
ssoares@ice.ufjf.br

[1] LApIC Research Group, Departament of Computer Science, Campus Universitário, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora 36.060-900, Brazil

[2] Department of Information Systems and Operations Management, Institute for Data, Process and Knowledge Management, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

[3] Departament of Computer Science, Campus Universitário, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora 36.060-900, Brazil

1 Introduction

Ontologies are computational models for representing objects in a domain and their relationships. Due to its high representative capacity and its level of formalism, the ontology model, which is human-readable and machine-readable, has been used in several applications to model the underlying concepts in a formal manner [1,2]. The ontology-based modeling is subjective and expert dependent; ontologies are built by people with distinct levels of expertise and goals. Therefore, concepts that describe the same type of object may be represented in different ways, both in syntax and structure, generating a problem of heterogeneity in data semantics.

The identification of semantically related concepts between different knowledge graphs (KGs) allows the development of broader applications and the execution of tasks that consume information from different sources, i.e., suggest properties to apply a KG join query [3] and return a set of tuples containing information from several sources.

The heterogeneity among various data sources is the main obstacle to the semantic interoperability. Finding mappings between entities of different ontologies contributes to the interoperability of systems, facilitating the exchange of information and, consequently, reducing the problem of semantic heterogeneity [4].

Ontology matching (OM) techniques can be used to find semantic correspondences between ontology elements with less human effort. The ontology matching is a NP-hard optimization problem [5] that can be addressed by several computational techniques. Due to the high heterogeneity of the ontologies, there is no technique that stands out from the others in all aspects [6]. Ontology meta-matching (OMM) approaches can be used in this scenario. An ontology meta-matcher approach uses methods to select the correct matching components to execute, and to adjust the multiple knobs (e.g., threshold, coefficients, weights, etc.) of the components that are part of the OMM process [7].

The literature suggests that the best results found in the meta-matching of ontologies are associated with the use of evolutionary algorithms [5,6]. The OMM process usually uses (1) multiple similarity measures, (2) a method of combining similarity measures, and (3) a method of selecting candidate matches, where evolutionary algorithms are commonly employed in at least one of these last two steps. Several evolutionary algorithms have been applied, and the most common are the Genetic Algorithm (GA) [8], the Memetic Algorithm [9], and the Particle Swarm Optimization (PSO) [10].

Since OMM process has several steps, components, and other factors that may impact the result, comparing different approaches can be a challenging task. The authors normally use the absolute numbers of Precision, Recall, and F-measure; however, using these three measures is not enough to understand the pros and cons of each approach. This is because each approach is composed of different similarity measures, combination methods and search methods, what makes it not clear which researchers' idea was responsible for the improvement in the final result. An isolated analysis of the contribution/cost of each algorithm can help researchers and developers to choose computational solutions that best suit their usage scenario. The objective of this manuscript is to present a series of analyses on the algorithms that are commonly used in each execution stage of an OMM tool. The analyses were made through experiments that were designed to verify the behavior of each algorithm in isolation, that is, varying the algorithms that are used in a specific step but maintaining the algorithms of the other steps. In this way, the results are obtained according to the algorithms that were part of the object of study, since the other stages of execution do not vary from one experiment to another.

The rest of the paper is organized as follows: Sect. 2 introduces the ontology meta-matching task, a brief overview of the methods used in this task, and the related works that aimed to compare different approaches to OMM, as well as our view of the problem and how the experiment can be divided and executed. Section 3 presents part of the research methodology, introducing the hypotheses that were tested in the work. Section 4 presents the framework that was used to conduct the experiments, describing how it fits to accommodate the proposals of this research. Section 5 finalizes the methodology, explaining the experiment models built within the framework for each hypothesis to be tested. In Sect. 6, we discuss the results and, finally, Sect. 7 presents the conclusions and limitations of the work.

2 Ontology meta-matching task

Based on the workflow presented by some authors [7,11], the ontology meta-matching task can be divided into steps. In the so-called Modeling step, authors usually describe which similarity measures are used to compare ontology entities. In the subsequent Optimization step, a combination method is defined to aggregate multiple similarity measures. Several authors present approaches based on finding the best set of parameters to aggregate multiple similarity measures [7], which makes the second step a prolific research topic [12,13]. After the Optimization step establishes the final confidence value for each candidate match, it is necessary to select the correspondences that have greater probabilities of correctness. Thus, in the third step, Correspondence Selection, heuristic search methods are usually used to define the correspondence set (alignment) closest to the optimum [4].

2.1 Popular methods used in OMM approaches

In this section, we describe the main techniques and algorithms used in the evaluation of this research, each one belonging to a specific step of the workflow.

Modeling is the step responsible for defining the similarity measures and creating the main structures consumed during the process. This research followed the classification presented in [14] to evaluate the most common similarity measures found in the literature. The main algorithms used in this step are the following:

- Levenshtein similarity measure is based on the number of single-character edits needed to transform one string into another. The premise is that the smaller the number of operations, the greater the similarity of the strings [15].
- A String Metric for Ontology Alignment (SMOA) is a similarity measure that addresses the similarity by analyzing the commonalities and the differences of the strings. The commonalities are based on the most common substring of the two inputs, and the differences are calculated based on the unmatched strings [16].
- Wu-Palmer is a linguistic similarity measure that uses the depths of the two synsets in the WordNet taxonomy, as well as the depth of the Least Common Subsumer (LCS), to determine relatedness [17].
- Similarity Flooding [18] is a taxonomy-based similarity measure who takes a set of correspondences as an input and uses these matches to propagate the similarity to the neighbor nodes, making use of the relationships network of the ontology to find new matches. The quality of the input set is essential to determine the final alignment; if the initial mapping contains wrong correspondences, the error will be propagated to the found correspondences.

String-based techniques like Levenshtein and SMOA are faster to compute compared to the others (WordNet linguistic techniques and the taxonomy-based Similarity Flooding); however, they don't analyze semantics or structural information of an entity, which leads to different sets of matches founded by different measures.

Optimization is the step responsible for combining different similarity measures and define the final confidence score for a candidate match. In this step, the literature shows that evolutionary metaheuristics are the most common algorithms used to execute this combination [19]. This research evaluated three different metaheuristics: Genetic Algorithm (GA), Prey–Predator Algorithm (PPA), and a Greedy Randomized Adaptive Search Procedure (GRASP). GA is based on the classical algorithm proposed by Holland [20] inspired by the evolutionary biology and implementation details can be found in [21]. PPA algorithm is based on the movement pressure that forces a set of preys (average solutions) to run away from a predator (worst solution); this algorithm was based on the approach proposed by [22] and implementation details can be found in [23]. Both GA and PPA are approaches for global optimization, to check the behavior of local search metaheuristics this research also evaluated a GRASP-based approach. GRASP-based approach consists of an interactive heuristic that tries to improve a specific solution at each iteration; full implementation is available in [24].

Every metaheuristic requires a way to assess the quality of a solution. There are several objective functions that can be used to guide the evolution of solutions, as follows:

- MatchFmeasure is a score that reproduces the behavior of the classical information retrieval score F-measure but without a reference set, that is, an unsupervised objective function. It is a harmonic mean between MatchRatio and MatchCoverage, where MatchRatio and MatchCoverage are substitutes for Precision and Recall, respectively [25].
- Linear System-based objective function evaluates the ability of a solution to solve a linear system built from a set of reference correspondences [21]. It is considered a semi-supervised function due to the size of the reference set, which is from 3 to 4% of the correct alignment. The main feature is time efficiency.

Correspondence Selection is the final step, with the goal to extract the final alignment from the highest confidence correspondences. There are several types of relationships between entities. In the context of this research, the goal is to find equality (1:1) relations. An equality relationship means that an entity of the source ontology can be equal to only one entity of the target ontology, which means that they both represent same real world concept. To evaluate the Correspondence Selection step, can be used different algorithms, such as the Hungarian algorithm [26] and Simulated Annealing (SA) [4]. The first one is a deterministic algorithm designed to solve the assignment problem in polynomial time. The second one is a metaheuristic based on a probabilistic local search. While the Hungarian algorithm finds a stable solution, the SA can be configured using different parameters, including the objective function.

2.2 Related works

Comparative studies on OMM approaches have evaluated the use of different meta-heuristics used in Optimization and Correspondence Selection steps. In [13], the authors analyzed multi-objective ontology matching approaches and aimed to evaluate the Optimization step. The authors make use of important metrics such as hypervolume, δ index, C metric, and others, which allow viewing the results from a statistical point of view. The main finding in [13] is the

better performance of the Optimal Multi-Objective Particle Swarm Optimization (OMOPSO) over other famous multi-objective algorithms such as NSGA-II, MOEA/D, and SPEA2.

Seeking to analyze the application of local search meta-heuristics in the Correspondence Selection step, in [27] the authors present a comparison on the TABU search, Simulated Annealing (SA), and other meta-heuristics. The experiments use the same set of similarity measures and integrate them in a weighted way through the OWA operator [28]. In [27], meta-heuristics are modeled to find the best candidate matches in one step after the aggregation of similarity measures; the goal is to find the alignment that minimizes the distance between entities (or maximizes the confidence value in the alignment). In addition to well-known metrics (Precision, Recall and F-measure), the authors [27] compared the convergence rate of each of the metaheuristics. The convergence analysis in this case is facilitated by the modeling chosen by the authors, since the solution space is composed of candidate alignments; therefore, it is possible to compare the modifications of the final solution in each iteration. The authors highlighted the TABU search performance over other meta-heuristics.

MELT [29] presents different tools to assess the similarity between the results of the matchers in order to promote different comparative analyses. MELT is a toolkit for ontology meta-matching whose components have been used individually in other researches [30]. The toolkit was designed to evaluate ontology matching tools as well as visualizing matching results. Its main characteristics are associated with the visualization of results and the integration with the tools provided by OAEI, such as SEALS and HOBBIT. The authors demonstrate the framework's evaluation methods by calculating similarity between the result of matchers who participated in Ontology Alignment Evaluation Initiative (OAEI) campaigns. The metric used in the article compares the similarity between correspondences found by different matchers, indicating the importance of qualitative analysis in the OMM problem.

Despite the advances in understanding the ontology meta-matching problem and approaches, there are some questions to be answered. New approaches have been published in the last years, but it is not clear whether the results are statistically significant and which new idea in the meta-matching process has resulted in better results. This occurs mainly due to the evaluation methodology adopted by the authors. In articles that compare different approaches, some authors [13] compare different meta-heuristics in the Optimization step and others [27] compare different meta-heuristics in the Correspondence Selection step, but when it comes to analyzing the impact of these algorithms on the solution, there is a gap. As the analysis is focused exclusively on meta-heuristics, there may be a false impression that the meta-heuristic was the main factor that contributed to the result. However, the OMM process is sequential and the result of one step directly affects the next step. Furthermore, the experiment is composed of similarity measures, objective functions, combination methods, and other factors that may have a greater influence on the quality of the result or on the time spent to obtain it.

In some articles that present new proposals for OMM, the evaluation methods focus on accuracy results and do not explore relevant characteristics of the algorithms used. In [10,31,32], different OMM approaches are proposed, and the evaluations are made in comparing Precision, Recall and F-measure values. These three metrics are sufficient to validate whether the experiment is capable of finding solutions to the problem, but they are insufficient, when applied in completely different experiments, to answer which part of the approach is responsible for the results. In [10], the authors evaluate ontologies synthetically created and there is no comparison with other approaches in the literature. In [32], the authors used the OAEI dataset and the results were compared to other approaches that were submitted to the same campaign using Precision, Recall, and F-measure values. Analyses that could bring more details of the results, however, are not easy to be done because there is a lack

of information regarding the behavior of the algorithms that are executed in each processing step, such as the confidence value of each similarity measure, or the cost of performing an objective function. In addition to the lack of information on parts of the process, there are matching approaches in the literature proposing different ways of representing measures [33] that have not yet been explored to match ontologies, increasing the range of open research in OM.

Unlike MELT, this article presents comparisons using not only the correlation rate but the efficiency and accuracy rates related to experiments with similar configurations to make clear the impact that a choice in the OMM workflow can cause to the results. Like [13,27], this study seeks to find answers that help to understand the role that each algorithm plays in the construction of the solution. We analyze the contribution of different similarity measures, different objective functions, and comparisons between methods that are not exclusively based on meta-heuristics. These answers may allow researchers to better understand the statistical significance of each computational solution.

3 Hypothesis definition

This section is dedicated to reporting the definition of each hypothesis tested in this work, as well as presenting the motivations and justifications behind each hypothesis. The first hypothesis concerns the use of multiple similarity measures and is defined as:

H1 To use String-based, Taxonomy-based and Linguistic Resources similarity measures is enough to create a resilient tool

Although similarity measures represent the basis for starting the experiment, most articles do not include similarity measures in their evaluation methodologies; they only describe which measures were used and how they work, as in [9,31,34]. It is difficult to determine which measures contributed to a particular alignment. The **H1** hypothesis tests the ability of different similarity measures to find different matches, using similarity measures from three different groups. Similarity measures can be classified with respect to the type of input they work with. A well-known classification is presented in [14]. In our experiments, we evaluated measures from three groups: String-based, Taxonomy-based, and Linguistic Resources. The three groups were chosen because they represent the majority of similarity measures that have been used by researchers in recent years. Data on the most frequent groups of similarity measures are shown in Fig. 1. Figure 1 shows the percentage that each group represented in articles published in the last 7 years; the groups are divided by the year of article publication and follow the classification of [14].

Despite String-based, Taxonomy-based, and Linguistic Resources are the most common types, there are other algorithms that can be used to assess the similarity between two objects, some of which may not have been used in OMM approaches. The knowledge domain represented by the ontology is a factor that can influence some authors to search for domain-specific similarity measures, for example, if the ontologies are in the health area, it may be interesting to look for a similarity measure that uses a medical thesaurus. Domain-specific measures can enhance the results but end up being restricted to just one application scenario. The normalized compression distance (NCD) is a distance measure that works for any type of data and has been used in clustering applications [35]. As far as we know, the NCD has not yet been tested on OMM tools. The **H2** hypothesis comes precisely to verify the applicability of the NCD.

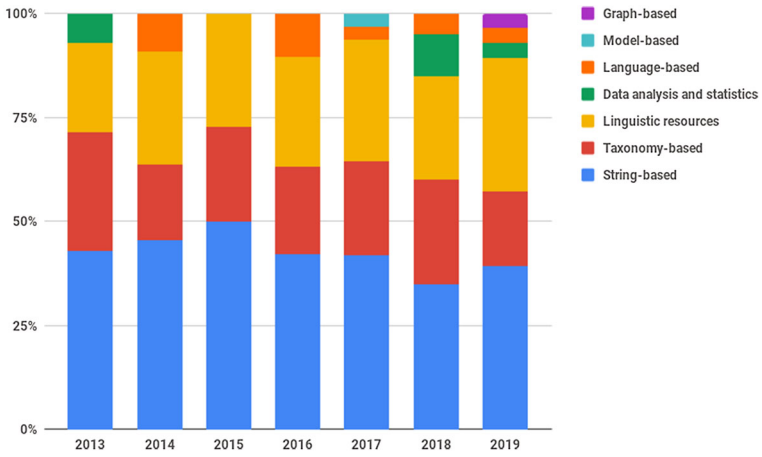


Fig. 1 Percentage of each type of similarity measure per year

H2 The use of the normalized compression distance can reduce the error rate of a ontology meta-matcher

After defining the hypotheses about the similarity measures used in the Modeling step, the next hypotheses refer to the meta-heuristics and objective functions, dividing between those used in the Optimization step and the ones used in Correspondence Selection step. There are several works addressing the OMM problem using different meta-heuristics, such as Genetic Algorithm (GA) [21], Memetic Algorithm [36], and Particle Swarm Optimization (PSO) [37]. Each meta-heuristic can make use of one or more objective functions, which are responsible for guiding the meta-heuristic solution(s) in the search space.

In this work, we divide the objective functions into supervised/semi-supervised and unsupervised approaches. Supervised approaches require a full reference alignment to compute their value, semi-supervised approaches require a relatively small reference alignment, and unsupervised approaches works independently, with no reference alignment required. Figure 2 presents a comparison between the number of supervised/semi-supervised and unsupervised approaches in the literature, pointing to an increase in the number of unsupervised in recent years. In the Optimization step, both supervised/semi-supervised [21] and unsupervised [36] approaches can be used. Both [21,36] evaluate the research in the OAEI bibliobenchmark, and the authors applied meta-heuristics in the optimization step, but with different objective functions. The evaluation of [21,36] is done in terms of the F-measure; [36] still includes comparisons of time and memory spent. Neither the articles that compare approaches [13,27], nor the articles that propose algorithms that fit in the Optimization step [21,36], present evaluation proposals about different objective functions, making it difficult to understand the contribution of each objective function and, likewise, making it difficult to study new trends, such as the data presented in Fig. 2. The **H3** hypothesis was built to evaluate two distinct objective functions, one supervised and one unsupervised.

H3 In the same experiment setup, an unsupervised approach can achieve results with quality close to that of a supervised one; however, the time for reaching this solution is longer

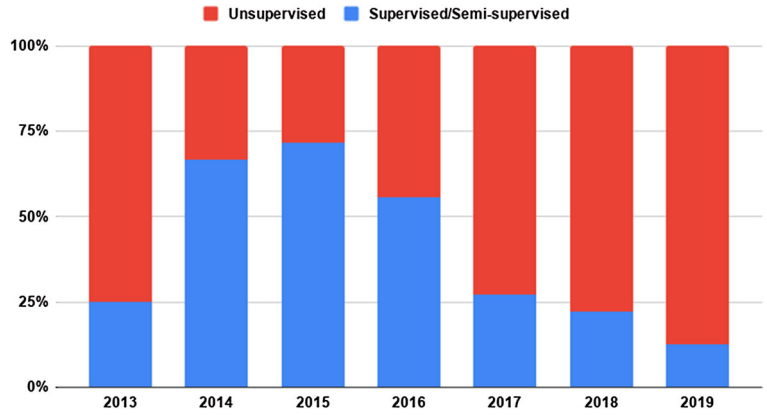


Fig. 2 Amount of unsupervised versus supervised/semi-supervised approaches per year

The last hypothesis was created based on the algorithms used in the Correspondence Selection step. There are several methods that can be used at this step [38], such as the Karp algorithm, Minimum Cost Flow (MCF), Hungarian Algorithm, or meta-heuristics [4]. Simpler deterministic algorithms like Hungarian optimize based only on the confidence value assigned to each relationship, while a meta-heuristic can evaluate other aspects such as the number of candidate matches for each entity. In [27], the authors compare different local search meta-heuristics that use the same objective function to search for optimal alignment. The evaluation is conducted using the F-measure score and the solution convergence rate. In [4], a tool named SANOM is proposed, and the authors use SA to search for optimal alignment. The solution used in SANOM fits in the Correspondence Selection step described in this work. SANOM was evaluated by numerous tracks from OAEI, demonstrating the versatility of the tool. The authors present analyzes of time consumption, Precision, Recall, F-measure, and other metrics.

Both works [4,27] present robust solutions for the correspondence selection step; however, some authors use simpler methods and still are able to find good solutions [21,38]. As the evaluation of [27] is carried out by exchanging one meta-heuristics for another and the SANOM evaluation is conducted comparing the final result of the experiment, there is no evidence of the effects of the Hungarian algorithm, since the articles focus on methods based on meta-heuristics. The **H4** hypothesis tests the cost/benefit of a correspondence selection algorithm based on meta-heuristics and another based on the Hungarian method.

H4 Matches via the selection based on Meta-heuristics can be computed in a execution time comparable to the Hungarian algorithm.

4 Ontology meta-matching experimentation framework

We make use of a framework to test ontology meta-matching approaches. It was built based on the division of the ontology meta-matching process into steps, where a set of computational solutions is available for each step. The framework covers the three main steps of the OMM process: the Modeling, Optimization, and Correspondence Selection steps (see Fig. 3). Each step constitutes a system module that includes algorithms to evaluate similarity between

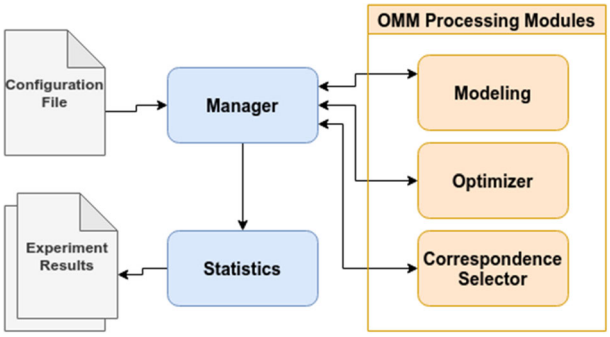


Fig. 3 System architecture

entities, optimization algorithms for combining similarity functions and selection algorithms to choose matches. Each module operates independently having its input data and a predefined output format. This architecture allows new approaches to be implemented at specific process steps facilitating the construction of different experiment models.

In Fig. 3, it is possible to observe the existence of a management module that controls the flow of execution of the experiment. This module is responsible for guaranteeing the execution and delivering the result of one execution step to the subsequent step. The processing modules are those that have different algorithms to perform the same OMM sub-task. Through changes in the configuration file, new experiment models can be created, processed, and evaluated under different measures of a statistics module. This framework is freely available[1] for researchers to use and improve it, and the following subsections describe in detail each of the processing modules.

4.1 Modeling

The modeling module is the first to be triggered by the manager module. This module has as input the files containing the pair of ontologies that will be aligned and the name of each similarity measure that will be used in the experiment. This framework was developed to process ontologies described in OWL or RDF. All this information is read and structured in computational objects that will be consumed in the following steps by means of the Jena API [39]. The output consists of objects containing the ontologies data and a set of similarity measures.

The user can implement his own similarity measures as well as use the measures available in the tool. Each similarity measure (matcher) generates a matrix with $|e_1|$ rows and $|e_2|$ columns, where e_1 and e_2 represent the set of entities from the source and target, respectively. Each cell in the matrix represents the confidence value assigned by that matcher for a specific candidate match. The user can compose a similarity measure or add new measures. In the first option, the code available allows the user to compose parts of similarity measures. The user can define a similarity measure by choosing its input and the metric used. For instance, it is possible to define a function that only compares ontology class labels and uses a certain edit distance metric, such as Levenshtein. The user function definitions are stored in a XML file stores and is read during runtime. To include new similarity measures, the user simply implements the Java interface that best suits the data. For example, a String-based measure

[1] https://bitbucket.org/nicolasferranti/heuristicontologymatching/src/master/.

Table 1 Similarity measures

Name	Classification	References
Wu palmer	Linguistic-based	[17]
Lesk	Linguistic-based	[41]
Similarity flooding	Taxonomy-based	[18]
JaroWinkler	String-based	[42]
Levenshtein	String-based	[15]
DamerauLevenshtein	String-based	[43]

[40] can be added using the *IEditDistance* interface because it consumes two strings, while functions such as Similarity Flooding [18] must use the *ISimilarityFunction* interface because it is necessary to use other information from the concept. Table 1 shows some of the measures available and their classification according to [40].

4.2 Optimization

The optimization step is the experiment step responsible for combining multiple similarity measures and defining a confidence value for each pair of ontology entities. An optimizer can be defined as a tuple $\sigma = (o_1, o_2, Sm, HSA)$ where:

- o_1 and o_2 are the computational structures of the ontologies to be matched, represented by an instance of type Model that contains the methods to retrieve the pertinent information of the ontology data.
- Sm represents the similarity measures set. For each $sim \in Sm, sim(Entity_1, Entity_2) = Conf_{sim}$ where $Entity_1, Entity_2$ are entities from distinct ontologies and $Conf_{sim} \in [0, 1]$.
- HSA is an heuristic search algorithm.

To perform the optimization process, it is common in the literature to use metaheuristics such as Genetic [7], Memetic [44], PSO [10] algorithms, and many other approaches as search algorithm. Evolutionary algorithms are becoming the state of the art in the search for solutions to this problem [45]. These search heuristics are usually used to find the best combination of similarity measures. Despite having inspiration in different phenomena, the sequence of operations of the meta-heuristics are similar, generally making use of (1) a phase of building one or more solutions, (2) diversification of solutions to explore the search space, and (3) intensification to improve the quality of existing solutions. The great advantage of using meta-heuristics as optimization algorithms is the level of abstraction that this solution has, each type of strategy must be adapted based on the characteristics of the problem, that is, it is a malleable solution whose effort will depend on the objectives of the researcher.

The framework allows the researcher to implement his own optimization algorithm as long as it respects the input and output formats of the execution. The interface developed for the implementation of an optimizer is divided into three levels of specialization. At the most abstract level, the user simply implements a function that returns a similarity matrix M containing the candidate matches and their confidence rates. At the intermediate level, the user has the inclusion of a Model object that delivers the methods of accessing the data of the ontologies. At the third level, the user can include in his optimization algorithm an objective function created based on a fitness interface. As the optimization algorithms, the framework provides a generic interface allowing the researcher to implement his own objective function

Table 2 Objective functions available

Name	Classification	References
Precision	Supervised	[46]
Recall	Supervised	[46]
F-measure	Supervised	[46]
Linear system based	Semi-supervised	[21]
Confidence sum	Unsupervised	[47]
MatchCoverage	Unsupervised	[48]
MatchRatio	Unsupervised	[48]
MatchFmeasure	Unsupervised	[48]

or use one (or more) of those available. To implement a new objective function, the interface requires the researcher to define the fitness calculation method and other simple characteristics regarding the function, for example, if the objective is to maximize or minimize.

To exemplify the use, three optimization approaches are available in the framework: a genetic algorithm (GA) , a prey–predator algorithm (PPA), and a greedy randomized adaptive search procedure (GRASP). Each of the three algorithms was implemented at the third level of specialization, that is, the algorithms implement a function that returns the similarity matrix M, use the Model object, and use an objective function based on the Fitness interface.

Several objective functions can be found in the literature. In this research, we classified them into supervised and unsupervised approaches in order to simplify the understanding of how these functions can work. A supervised objective function requires a reference alignment to evaluate the quality of a solution, while the unsupervised function works independently. Table 2 presents the set of objective functions available in the framework, as well as the classification we propose for the type of approach and the reference where more details of the algorithm can be found. The objective function based on the linear system was classified as semi-supervised because it uses a relatively small reference alignment when compared to the others supervised. The objective functions guide the search performed by the meta-heuristic in the solution space. At the end of the execution, the meta-heuristic returns the parameters to aggregate multiple similarity measures. These parameters are used to build a unified similarity matrix which represents the Optimization module output. This similarity matrix contains all possible candidate correspondences and their similarity score. Let $E_{1,i}$ be the i_{th} entity of O_1 and $E_{2,j}$ be the j_{th} entity of O_2, where $i \in \{1, |O_1|\}$ and $j \in \{1, |O_2|\}$. Each row of the matrix M has a tuple $(E_{1,i}, E_{2,j}, \eta_{i,j})$ where $\eta_{i,j}$ is the confidence value for the candidate match $(E_{1,i}, E_{2,j})$.

4.3 Correspondence selection

The correspondence selection module is responsible for selecting the candidate matches of matrix M that are most likely to be a correct match. The algorithms used in this step vary according to the researcher's objective. The size of the ontologies also influences the choice of correspondence selection algorithms since the weights adjustment process of the optimizer is costly for large-scale ontologies. What is observed in the literature is that authors who work with large-scale ontologies allocate effort in the stage of correspondence selection, making use of more complex algorithms such as memetics [49], NSGA-III [50], and others [51,52]. While authors who work with conventional ontologies and focus on the optimization process

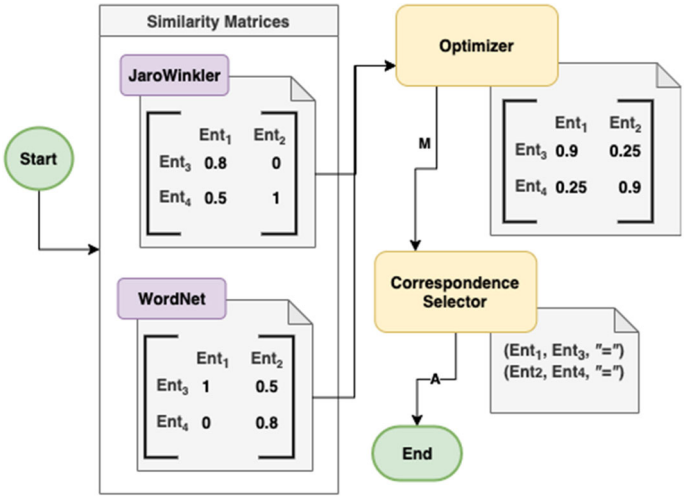


Fig. 4 Data flow

use simpler selection algorithms such as greedy heuristics and other deterministic algorithms [21,47]. This framework allows the researcher to implement their own selection algorithms through a standard interface, whether they are specific heuristics or meta-heuristics. To propose a new solution, the researcher needs to implement a method that uses the M similarity matrix as an input and returns a set of matches that represents the final alignment.

To exemplify this process, Fig. 4 shows the data flow starting from the confidence value calculated by each matcher for each candidate correspondence. If e_1 and e_2 are entities in the source ontology and e_3 and e_4 are entities in the target ontology, the experiment optimizer combines the confidence value while the match selector chooses the most reliable pairs. This framework provides three selection algorithms as examples. The first algorithm is an iterative greedy algorithm that seeks, at each iteration, the highest $\eta_{i,j}$ and selects the most reliable match. The second algorithm is based on the Hungarian method and details are presented in [26]. The third algorithm presents a Simulated Annealing metaheuristic implementation that, unlike the others, makes use of a structure to represent a solution based on correspondences. This structure is available for consumption in other algorithms and consists of a list of objects, each one containing the information about a single correspondence.

The algorithms available in the framework work with cardinality rates (1:1), which means each entity can only be aligned once with another entity. Therefore, since two entities are chosen as a match, all other candidate matches that have one of these entities must be removed from matrix M. Other cardinalities can be added. The output format of this module is an alignment (A) that contains the most reliable correspondences in M.

5 Experimental planning

The H1 hypothesis concerns the use of multiple similarity measures to verify the ability of finding correspondences for each type of measure. Groups were created based on the type of input that classifies the measure. Based on the classification model [40], four groups were created that form the basis for four different experiment models presented in Fig. 5.

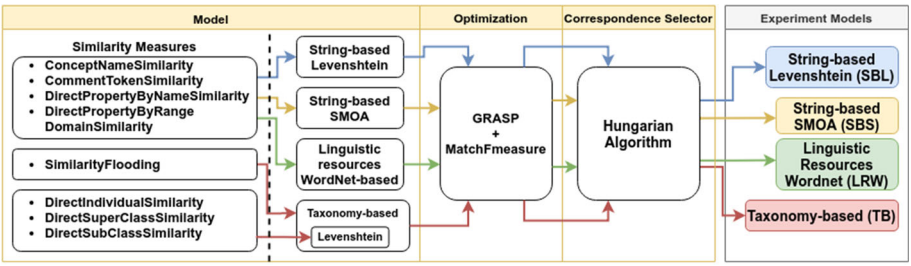


Fig. 5 Configurations created for evaluation of H1

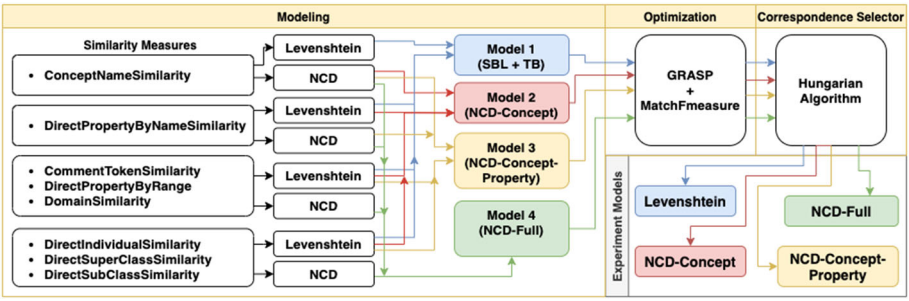


Fig. 6 Configurations created for evaluation of H2

The first two models (SBL and SBS) represent terminological measures operating with two different similarity measures: Levenshtein distance (used in [53–55]), and SMOA [56] which according to [16] is the most performing measure for the OM problem. The third model (LRW) is also terminological, but adopts a linguistic approach based on WordNet. Finally, the last model (TB) includes measures at the structural level, represented mainly by the similarity flooding algorithm which is a Taxonomy-based approach.

The algorithms used in the Optimization and Correspondence Selection steps were chosen empirically in order to reduce stochastic factors that impact both the quality of the result and the time spent to obtain it. The main objective was to provide that the data collected from the experiment were as less dependent as possible on the Optimization and Correspondence Selection steps. It was used the Hungarian method (deterministic method), the MatchFmeasure (not influenced by reference alignments), and GRASP.

Hypothesis H2 is also associated with the Modeling module. Four different experiment models were built: Levenshtein, NCD-Concept, NCD-Concept-Property, and NCD-Full (see Fig. 6). The first one (Levenshtein) consists of the combination of similarity measures that were separated in H1, all using the Levenhtein distance (Model 1) as similarity calculation strategy. The same set of measures was used in the second experiment model (NCD-Concept), however, with the NCD replacing the Levenshtein distance in the calculation of similarity between concept names (Model 2). In the NCD-Concept-Property model (Model 3), in addition to the concept name similarity, the NCD was also applied to property names. Finally, in the NCD-Full model (Model 4), all similarity measures are configured to use the NCD. The objective is to assess the impacts of the NCD in this scenario. As in H1, the Optimization and Correspondence Selection algorithms were chosen to minimize stochastic variations both in time and in the quality of the result so that the experiment actually portrays the behavior of similarity measures.

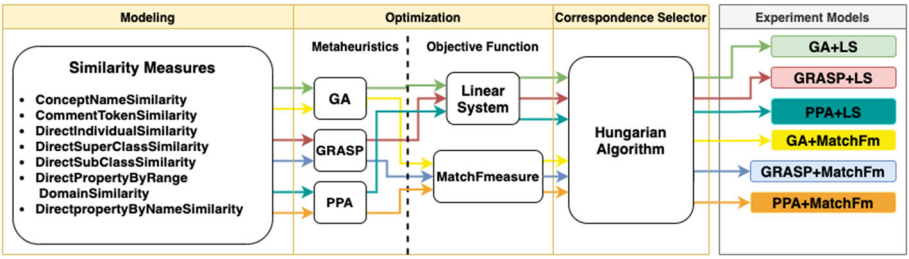


Fig. 7 Configurations created for evaluation of H3

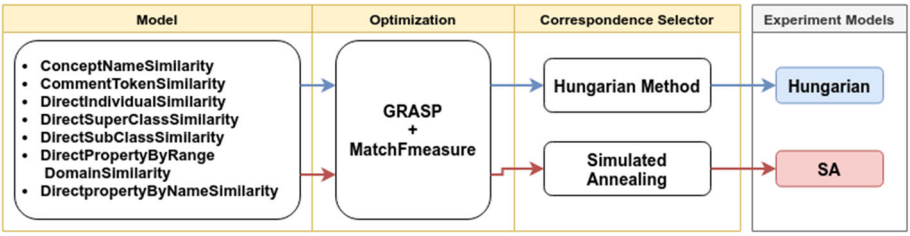


Fig. 8 Configurations created for evaluation of H4

The H3 hypothesis is associated with the use of different objective functions. To test it, two objective functions of the framework were selected, both of them can be used in the same application scenario, where a candidate alignment is sought, which will be validated later by a specialist. The objective functions are: maximizing the MatchFmeasure (unsupervised) and minimizing the error when solving the linear system (semi-supervised). Both objective functions were tested with three different meta-heuristics: GA, GRASP, and PPA. The details of each model are shown in Fig. 7. The Hungarian algorithm was used as a correspondence selector by the H3 experiment models in order to minimize the stochastic factors and allow the result to occur due to the variation in the objective and meta-heuristic functions in the optimization step. Also, the set of similarity measures is composed of measures that until then were separated in hypotheses H1 and H2 but which presented promising results when combined.

The H4 hypothesis suggests the comparison between different algorithms that can be executed in the correspondence selection step, with an emphasis on the Hungarian method [26] and methods based on heuristics. To test it, two experiment models were created, both with the same algorithms in the Modeling and Optimization steps but varying the Correspondence Selection algorithm. In Fig. 8, the Hungarian model uses the Hungarian method as a correspondence selector while the SA model uses an implementation of the Simulated Annealing algorithm based on the proposal of [4].

6 Experimental results

Our experiments were carried out using one of the datasets provided by the Ontology Alignment Evaluation Initiative[2] (OAEI). Among the benchmarks made available by OAEI, the bibliobenchmark is the one that better suits the ontology meta-matching problem. Each test

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

Table 3 F-measure rates for the four experiment models

#Test	Precision	Recall	F-measure
String-based Levenshtein (SBL)			
1xx	1.000	1.000	1.000
2xx-1	0.923	0.947	**0.931**
2xx-2	0.238	0.268	0.248
3xx	0.808	0.807	**0.804**
String-based SMOA (SBS)			
1xx	1.000	1.000	1.000
2xx-1	0.897	0.921	0.905
2xx-2	0.236	0.268	0.247
3xx	0.755	0.757	0.754
Linguistic resources wordnet (LRW)			
1xx	1.000	1.000	1.000
2xx-1	0.894	0.918	0.903
2xx-2	0.245	0.276	0.255
3xx	0.652	0.660	0.653
Taxonomy-based (TB)			
1xx	1.000	1.000	1.000
2xx-1	0.869	0.893	0.877
2xx-2	0.336	0.379	**0.351**
3xx	0.595	0.596	0.594
Composition of SBL + TB			
1xx	1.000	1.000	1.000
2xx-1	0.944	0.967	0.952
2xx-2	0.407	0.452	0.423
3xx	0.808	0.807	0.804

Values in bold refer to the highest value by test group among the four primary similarity measures groups

case present in the bibliobenchmark consists of the alignment between a source synthetic ontology and a target ontology. The source ontology is the same in each test case, while the target ontology varies according to the test number. In test ranges 1xx and 2xx, the target ontology is created systematically from the source ontology where, at each test, part of the ontology information is discarded/modified in order to evaluate how the algorithm behave when this information is lacking. In range 3xx, the alignment between the source ontology and real ontologies in the domain of bibliographic references is verified. This test base is suitable for meta-matching because what is expected from this type of approach is an adaptive behavior that rewards the most efficient solutions to maximize the quality of the result in different scenarios.

The dataset was divided into test subsets. The 1xx subset is simpler than the others, as it contains only generalization or language restriction misrepresentations. The 3xx subset contains all tests with real ontologies. Tests 2xx, which contains misrepresentations related to hierarchy, instances, nomenclature, classes, and others, was subdivided into two subsets. The subset 2xx-1 contains all tests that have one or two misrepresentations, while subset 2xx-2 contains test cases that have more than two misrepresentations. All tests were conducted

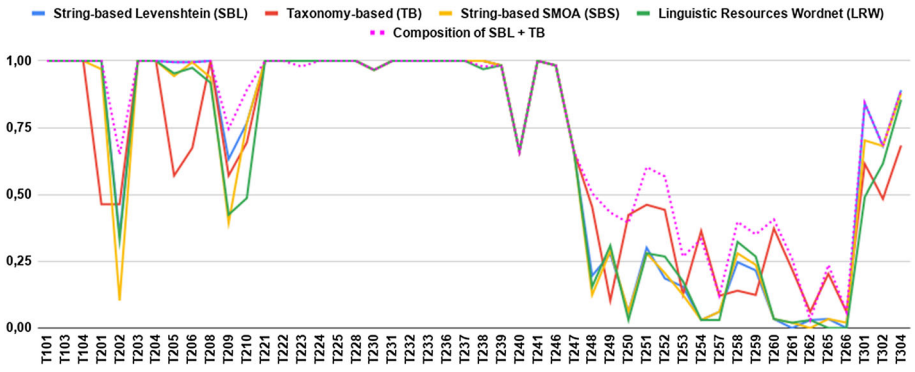


Fig. 9 Best F-measure rate for each H1 model, for each test case

by an Ubuntu 14.04.5 LTS machine with Intel Xeon CPU E5-2650 v2 @ 2.60GHz and 16GB RAM.

6.1 H1 evaluation

As mentioned before, H1 hypothesis concerns the use of multiple similarity measures to verify the ability of finding correspondences for each type of the most common similarity measure groups: String-based, Linguistic Resources, and Taxonomy-based (see Fig. 1). Table 3 shows the accuracy by subset of each model built for the H1 test, in the 1xx range all models achieved the maximum result, while in the others the best results were with the String-based Levenshtein (SBL) and Taxonomy-based (TB) models. Based on this preliminary result, a model composed of measures from the SBL and TB models was created; the results of this model are also presented in Table 3. The results achieved by the SBL + TB model equal or exceed the others in terms of F-measure. In some test cases, such as T258 and T259 in Fig. 9, neither SBL nor TB achieves the best result, but the combination of both allowed new correspondences to be found. Tables 4 and 5 show the Jaccard coefficient between the SBL, TB, and SBL + TB models for the T209 and T258 tests, respectively. The higher the Jaccard coefficient, the more correspondences the experiments found in common, whether they were correct or not. In the T209 test, SBL achieved the best results followed by TB, the SBL + TB model was able to surpass the others in T209, and, according to the Jaccard coefficient, the matches found by SBL + TB are more similar to those of TB, which indicates a greater contribution by TB measures even SBL having been higher than TB in this case. In the T258 test, where neither SBL nor TB did well, but SBL + TB outperformed the others, the Jaccard coefficient shows that new matches were found, since the correlation between the experiments is low.

In terms of execution time, the composite model of SBL + TB was also more expensive than the others, since it has a larger set of similarity measures. Figure 10 shows the mean execution time in seconds of each model for each test case, the LRW model ends up spending more time compared to the other terminological models because it requires an external connection to search on WordNet. The model composed by SBL + TB spent an average of 37 s, while the String-based Levenshtein (SBL) model and the Taxonomy-based (TB) model spent an average of 21 and 17 s, respectively.

Table 4 Jaccard coefficient for H1 test number 209

Jaccard #209	SBL	TB
SBL + TB	0.500	0.803
SBL	1.000	0.422

Table 5 Jaccard coefficient for H1 test number 258

Jaccard #258	SBL	TB
SBL + TB	0.155	0.069
SBL	1.000	0.075

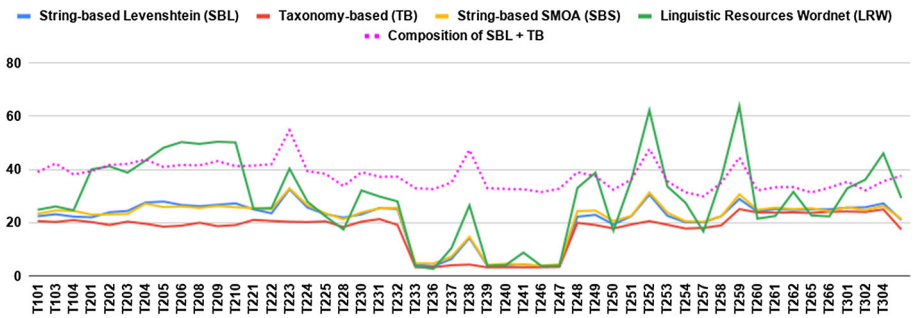


Fig. 10 Mean runtime in seconds of each model for each test case

To compare the results achieved by the composite model of SBL + TB with results from the literature, two studies were selected whose evaluation uses the same OAEI benchmark and the results are divided by the test number. Table 6 shows the results made available by articles [53,57], as well as the result achieved by the SBL + TB model for the same test cases. In this case, the H1 hypothesis proves to be true, because when using String-based, Taxonomy-based, and Linguistic Resources measures in an isolated environment, which does not suffer variations in the algorithms of other processing steps, it was possible to achieve results compatible with those in the literature. The studies pointed to different groups of correspondences associated with each set of similarity measures that, when combined, allow the creation of a more resilient tool, albeit slower.

6.2 H2 evaluation

Hypothesis H2 tests the use of NCD as a new similarity measure applied in OMM. Table 7 shows the accuracy results of the models that evaluate the NCD; it is also possible to observe in how many tests, of each specific range, a model achieved the best result among all models (last column). The results of the NCD varied, in the 3xx test range that includes alignments with real ontologies, there was no result capable of surpassing the Levenshtein model (SBL + TB), and however, in the 2xx-1 and 2xx-2 ranges there were some test cases whose result was higher, mainly in the 2xx-2 range of the NCD-Concept-Property model, where eight test cases were better than the others. Despite the improvement in some specific test cases, the average value points to compensation in the other cases causing the difference to appear in the third decimal place. Figure 11 shows that the use of NCD in some of the similarity

Table 6 Comparison of SBL + TB with literature approaches in terms of F-measure

Test	[57]	[53]	SBL + TB
101	1.000	1.000	1.000
103	1.000	1.000	1.000
104	1.000	1.000	1.000
201	0.940	0.940	**1.000**
203	0.990	0.990	**1.000**
204	0.980	0.980	**1.000**
205	0.930	0.930	**0.995**
206	0.700	0.700	**0.995**
221	1.000	1.000	1.000
222	1.000	1.000	1.000
223	**0.990**	**0.990**	0.979
224	1.000	1.000	1.000
225	1.000	1.000	1.000
228	1.000	1.000	1.000
230	**1.000**	**1.000**	0.966
231	1.000	1.000	1.000
301	0.750	0.810	**0.842**
302	0.740	**0.850**	0.681
304	**0.930**	**0.930**	0.890
Mean	0.945	0.954	**0.966**

In bold is the highest value by test case

measures did not impact on large differences in the experiment execution time, in the first half of the test cases the intermediate models obtained better performance than Levenshtein, a situation that is reversed from the test T258 ahead. In general, the NCD demonstrated that it can be a metric to be explored in OMM approaches; the H2 hypothesis was found to be true in some of the test cases but not in the vast majority.

6.3 H3 evaluation

In hypothesis H3, three meta-heuristics and two distinct objective functions were evaluated, totaling six experiment models. Table 8 shows the accuracy result of each model, of the models that used the function based on the linear system (LS), GRASP presented better results in the ranges 2xx-1 and 3xx while GA and PPA were better in the range 2xx-2. The replacement of the LS function by MatchFm resulted in a jump in the quality of the result, mainly for the GRASP and PPA models, demonstrating that this objective function alone has the potential to show new points in the solution space. Among the models that used MatchFmeasure, GRASP and PPA obtained, on average, better results than GA, where GRASP was better in the 3xx range and the PPA was better in the 2xx-2 and 2xx-1 ranges. However, Fig. 12 shows that PPA + MatchFm was much more expensive than GRASP + MatchFm in the vast majority of test cases, spending more than twice the time, which shows that in the experiments conducted GRASP had the best cost/benefit between the approaches that used MatchFm. Despite showing improvement in the quality of the results, the experiments that used MatchFm were more costly in terms of execution time than the LS, spending

Table 7 F-measure rates for the four experiment models of H2

#Test	Precision	Recall	F-measure	*N*
Levenshtein				
1xx	1.000	1.000	1.000	0
2xx-1	0.944	0.967	0.952	2
2xx-2	0.407	0.452	0.423	5
3xx	0.808	0.807	**0.804**	3
NCD-concept				
1xx	1.000	1.000	1.000	0
2xx-1	0.943	0.967	0.951	3
2xx-2	0.411	0.457	0.426	7
3xx	0.800	0.798	0.796	2
NCD-concept-property				
1xx	1.000	1.000	1.000	0
2xx-1	0.933	0.957	0.941	1
2xx-2	0.413	0.461	**0.429**	8
3xx	0.753	0.756	0.752	0
NCD-full				
1xx	1.000	1.000	1.000	0
2xx-1	0.937	0.961	**0.954**	2
2xx-2	0.402	0.448	0.417	2
3xx	0.746	0.749	0.744	1

N represents the number of cases with the best result. Bold highlights the highest scores found per test range

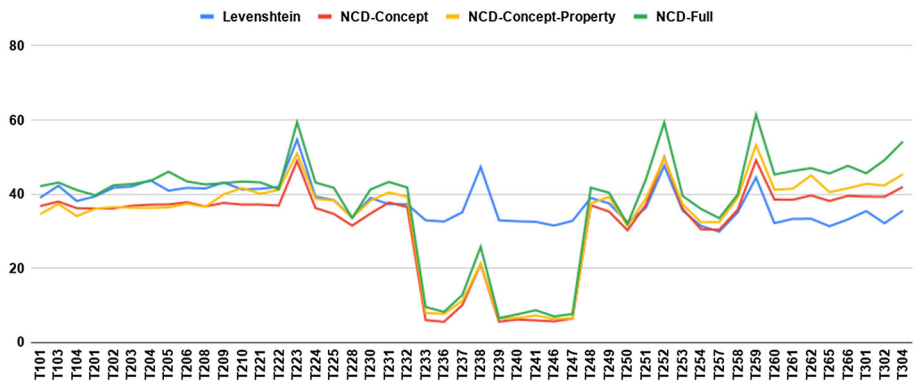


Fig. 11 Mean runtime (seconds) of each H2 model for each test group

more than twice the time. Therefore, if the execution time is a determining factor in the use of the tool, it may be worthwhile to select some input correspondences for the construction of the linear system. Although they can be obtained more quickly, the results of the models with LS vary more as pointed out in Figs. 13 and 14. Figures 13 and 14 show the mean and standard deviation for each test case, indicating that in most tests at switching from LS to MatchFm the stability of the experiment increased. Stability can be associated with the set of correspondences that are analyzed by the objective function, since MatchFm takes into

Table 8 F-measure rates for the six experiment models of H3

#Test	Precision	Recall	F-measure
GA + LS			
1xx	1.000	1.000	1.000
2xx-1	0.934	0.957	0.942
2xx-2	0.402	0.443	0.416
3xx	0.604	0.613	0.607
GRASP + LS			
1xx	1.000	1.000	1.000
2xx-1	0.940	0.964	**0.948**
2xx-2	0.358	0.398	0.372
3xx	0.799	0.800	**0.797**
PPA + LS			
1xx	1.000	1.000	1.000
2xx-1	0.925	0.948	0.933
2xx-2	0.399	0.443	**0.414**
3xx	0.547	0.556	0.550
GA + MatchFm			
1xx	1.000	1.000	1.000
2xx-1	0.939	0.963	0.948
2xx-2	0.353	0.395	0.367
3xx	0.796	0.796	0.793
GRASP + MatchFm			
1xx	1.000	1.000	1.000
2xx-1	0.944	0.967	0.952
2xx-2	0.407	0.452	0.423
3xx	0.808	0.807	**0.804**
PPA + MatchFm			
1xx	1.000	1.000	1.000
2xx-1	0.945	0.969	**0.954**
2xx-2	0.416	0.462	**0.432**
3xx	0.794	0.794	0.791

Highlights in bold for the highest values by test range grouped by objective function

account all possible pairs of entities while the LS models use a reference set that has 3–4% of the total correct correspondences.

The experiments show that each algorithm has its pros and cons, both an objective function and a meta-heuristic are capable of providing leaps in terms of quality and execution time, leaving the researcher to adapt them to their usage scenario. The H3 hypothesis was proven true for the tested algorithms because the quality of the solutions found as a result of MatchFm was close to the solutions found with LS models, surpassing the LS-based models in numerous test cases. In fact, it takes more time to find these solutions since there is no prior knowledge embedded in the objective function that limits the evaluation scope, which means that the unsupervised approach checks for more possibilities and spends more time.

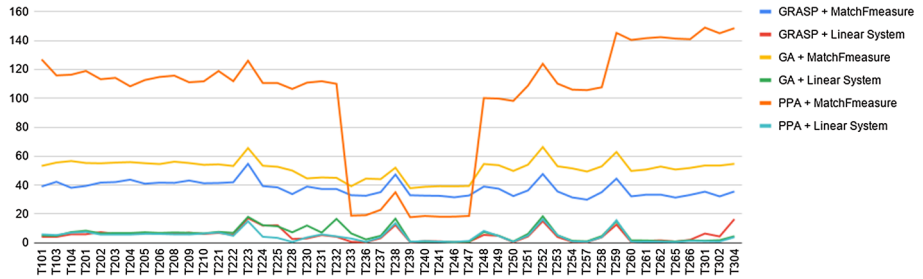


Fig. 12 Mean runtime (seconds) of each H3 model for each test case

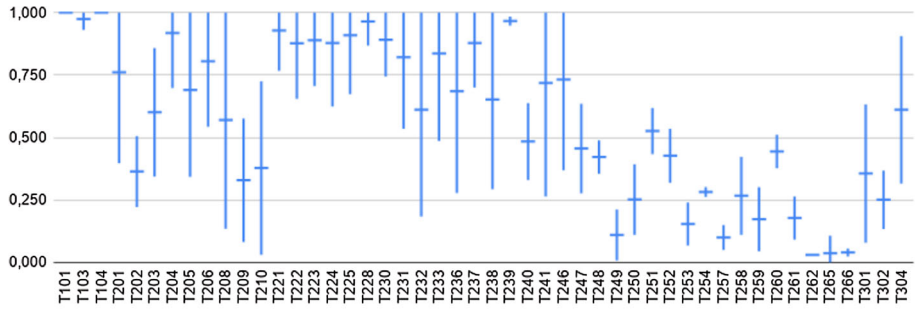
6.4 H4 evaluation

In H4, two algorithms for correspondences selection were evaluated: the Hungarian method and SA metaheuristic. Table 9 presents the Precision, Recall, and F-measure averages for each test range. The two models showed similar results, the Hungarian method was slightly better in the 2xx-1 and 2xx-2 ranges while the SA was better in the 3xx range. Table 9 also counts the number of tests for each range where one model was better than the other, demonstrating that there was a lot of alternation in the dominance of the 2xx-2 range and that although the SA obtained better results in more tests in this range, the difference in cases where the Hungarian method dominated was greater since the mean of the SA was lower. The results of Table 9 show that the stochastic factors related to heuristics allow new correspondences to be found in the same way that it is possible to lose other correspondences.

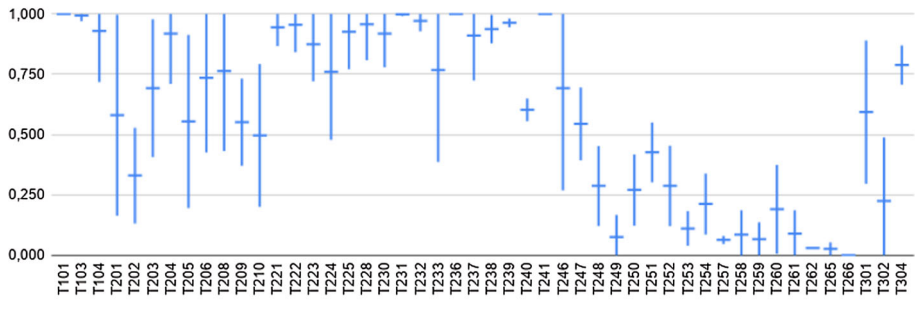
Figure 15 shows the mean execution time spent by each model in each test case, in the total mean, SA spent approximately 35 s while Hungarian spent approximately 38 s per test. The data in Fig. 15 point to similar results in the first test cases and greater variations in the second and third half. When analyzed together, the data in Table 9 and Fig. 15 show that algorithms with approximate solutions such as SA have the potential to achieve results close to optimal algorithms such as Hungarian. Figure 15 points to similar times in several test cases, but heuristic algorithms like SA have parameters that can be configured to reduce processing time, which can be useful in matching large-scale ontologies, where the number of entities is significantly higher than those in the OAEI bibliobenchmark. The heuristic algorithms also have the advantage of being able to work with different objective functions that can embed other parameters to be optimized, while the Hungarian method works in a more restricted way using the final confidence value that is assigned to each candidate correspondence. The H4 hypothesis proved to be true, since there were test cases where the SA obtained better results than the Hungarian and in a similar time, which can still be improved through the optimization of the SA configuration parameters.

7 Concluding remarks

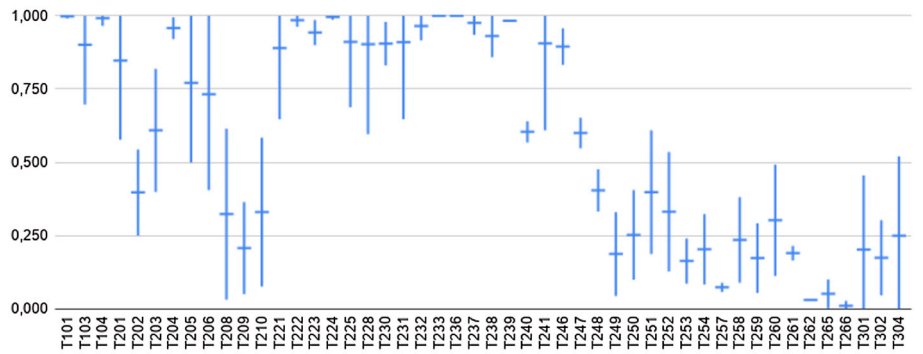
This manuscript presented an analysis of algorithms commonly used in the processing steps of OMM approaches. The main objective was to evaluate the cost/benefit of the algorithms tested in a controlled environment, that is, to compare experiments with small modifications in the same processing step. This methodology allowed to identify the contribution of each algorithm used in different parts of a compound solution to the problem of OMM.



(a) GA+LS



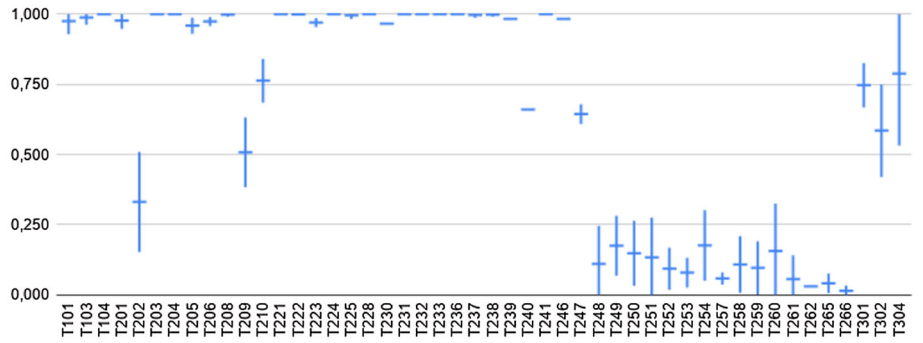
(b) GRASP+LS



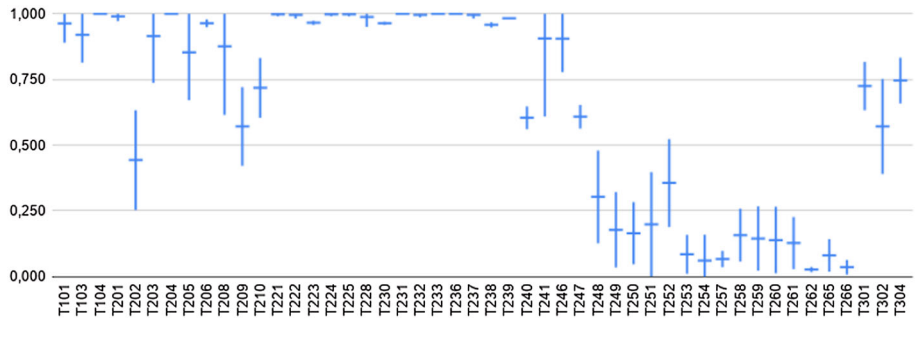
(c) PPA+LS

Fig. 13 Mean F-measure and deviation of H3 linear system based experiment models

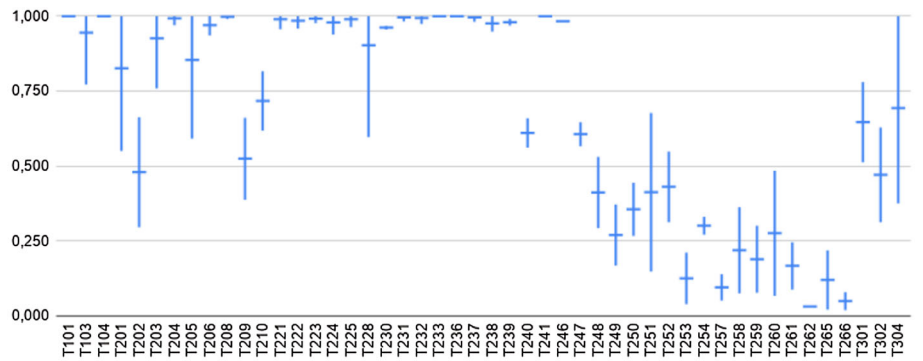
The results can be divided by the three processing steps: Modeling, Optimization, and Correspondence Selection. In the Modeling stage, it could be verified that the more characteristics of an entity can be captured by a similarity measure, the better the quality of the tool, with emphasis on the results achieved by taxonomic similarity measures and string-based measures using the Levenshtein distance. The results showed that even if two sets of similarity measures do not achieve good results when used individually, it does not mean that they should be discarded, since their combination can provide new correspondences. Still on the



(a) GA+MatchFm



(b) GRASP+MatchFm



(c) PPA+MatchFm

Fig. 14 Mean F-measure and deviation of MatchFm based experiment models

Modeling stage, experiments with the NCD reaffirmed that there is no technique that stands out from the others in all cases, since there was improvement in some cases and worsening in others, which makes the option of using the metric be left to the researcher based on the characteristics of the ontologies in alignment.

In the optimization stage, the results showed that algorithms with a focus on local optimization have the potential to find good solutions to the problem, as was the case with GRASP

Table 9 F-measure rates for the two experiment models of H4

#Test	Precision	Recall	F-measure	*N*
Hungarian				
1xx	1.000	1.000	1.000	0
2xx-1	0.944	0.967	**0.952**	**3**
2xx-2	0.407	0.452	**0.423**	6
3xx	0.808	0.807	0.804	0
SA				
1xx	1.000	1.000	1.000	0
2xx-1	0.939	0.963	0.947	2
2xx-2	0.399	0.444	0.414	**7**
3xx	0.812	0.811	**0.808**	**1**

N represents the number of cases with the best result. Bold highlights the highest score/number of test cases with best score

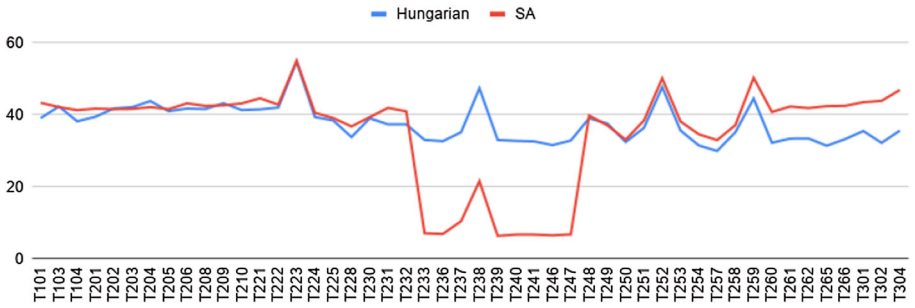


Fig. 15 Mean runtime (seconds) of each H4 model for each test case

that surpassed the others in several cases and with different objective functions. Of the objective functions studied, MatchFmeasure showed results with higher quality and stability in multiple executions; however, the function based on the linear system is faster, spending approximately less than half the time of MatchFmeasure. In the correspondence selection stage, it was seen that both deterministic and heuristic methods can find good solutions to the problem, while deterministic ones present stable solutions, heuristics have variations that can save processing time and find different groups of correspondences.

A tool for experimenting with OMM approaches was built to conduct this study and is available for future research,[3] this tool can be configured with new algorithms to test other research questions. The experiments conducted in this work used only mono-objective objective functions; however, the tool architecture supports and facilitates the creation of multi-objective approaches, opening a new range of options that allows the construction of more robust algorithms with greater ability to visualize the solution space.

The results of this study were obtained using the OAEI bibliobenchmark. OAEI has several other evaluation bases to test OM approaches and some characteristics of these bases can influence the result of the studied algorithms. One of the datasets available at OAEI concerns the alignment of large-scale ontologies, which increases the difficulty due to the large volume of data; this factor may make the use of some of the methods studied in this

[3] https://bitbucket.org/nicolasferranti/heuristicontologymatching/src/master/.

research unfeasible. In future works, it is intended to extend the analysis to other bases, verifying the variation in results.

Funding Open access funding provided by Vienna University of Economics and Business (WU).

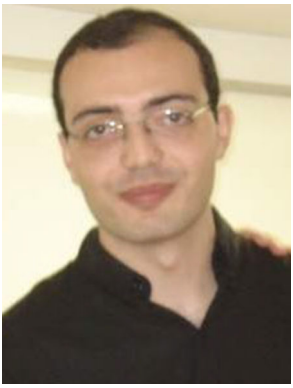
References

1. Bandrowski A, Brinkman R, Brochhausen M, Brush MH, Bug B, Chibucos MC, Clancy K, Courtot M, Derom D, Dumontier M et al (2016) The ontology for biomedical investigations. PLoS ONE 11(4):e0154556
2. Hoehndorf R, Schofield PN, Gkoutos GV (2015) The role of ontologies in biological and biomedical research: a functional perspective. Brief Bioinform 16(6):1069–1080
3. Bergami G, Magnani M, Montesi D (2017) A join operator for property graphs. In: EDBT/CDT workshops
4. Mohammadi M, Hofman W, Tan YH (2019) Simulated annealing-based ontology matching. ACM Trans Manag Inf Syst (TMIS) 10(1):3
5. Kureychik V, Semenova A (2017) Combined method for integration of heterogeneous ontology models for big data processing and analysis. In: Computer science on-line conference. [S.l.]. Springer, pp 302–311
6. Xue X, Tang Z (2017) An evolutionary algorithm based ontology matching system. J Inf Hiding Multimed Signal Process 8(3):551–556
7. Xue X, Pan JS (2018) A compact co-evolutionary algorithm for sensor ontology meta-matching. Knowl Inf Syst 56(2):335–353
8. Gulić M, Vrdoljak B, Ptiček M (2018) Automatically specifying a parallel composition of matchers in ontology matching process by using genetic algorithm. Information 9(6):138
9. Ramesh M, Karthikeyan IP, Meenachi IDNM, Baba MS (2016) Optimizing ontology alignment for nuclear information system. Int J Emerg Technol Eng Res 4:6–9
10. Semenova AV, Kureychik VM (2016) Multi-objective particle swarm optimization for ontology alignment. In: 2016 IEEE 10th international conference on application of information and communication technologies (AICT). [S.l.]. IEEE, pp 1–7
11. Silva MF, Baião FA, Revoredo K (2014) Towards planning scientific experiments through declarative model discovery in provenance data. In: 2014 IEEE 10th international conference on e-science. [S.l.], vol 2. IEEE, pp 95–98
12. Otero-Cerdeira L, Rodríguez-Martínez FJ, Gómez-Rodríguez A (2015) Ontology matching: a literature review. Expert Syst Appl 42(2):949–971
13. Acampora G, Ishibuchi H, Vitiello A (2014) A comparison of multi-objective evolutionary algorithms for the ontology meta-matching problem. In: 2014 IEEE congress on evolutionary computation (CEC). [S.l.]. IEEE, pp 413–420
14. Shvaiko P, Euzenat J (2013) Ontology matching: state of the art and future challenges. IEEE Trans Knowl Data Eng 25(1):158–176
15. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Doklady 10(8):707–710
16. Stoilos G, Stamou G, Kollias S (2005) A string metric for ontology alignment. In: International semantic web conference. [S.l.]. Springer, pp 624–637
17. Wu Z, Palmer M (1994) Verbs semantics and lexical selection. In: Association for computational linguistics. Proceedings of the 32nd annual meeting on Association for Computational Linguistics. [S.l.], pp 133–138
18. Melnik S, Garcia-molina H, Rahm E (2002) Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings 18th international conference on data engineering. [S.l.]. IEEE, pp 117–128

19. Ferranti N, Soares SSRF, Desouza JF (2021) Metaheuristics-based ontology meta-matching approaches. Expert Syst Appl 173:173114578
20. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3:95–99. https://doi.org/10.1023/A:1022602019183
21. de Souza JF, Siqueira SWM, Nunes B (2019) A framework to aggregate multiple ontology matchers. Int J Web Inf Syst 16(2):151–169
22. Tilahun SL, Ong HC (2015) Prey–predator algorithm: a new metaheuristic algorithm for optimization problems. Int J Inf Technol Decis Mak 14(06):1331–1352
23. Ferranti N, de Souza JF, Soares SSRF (2021). A prey–predator approach for ontology meta-matching. J Data Seman 1–12. https://doi.org/10.1007/s13740-021-00125-y
24. Ferranti N, Mouro JR, Mendonça FM, de Souza JF, Soares SSRF (2021) A framework for evaluating ontology meta-matching approaches. J Intell Inf Syst 56(2):207–231
25. Xue X, Wang Y (2015) Optimizing ontology alignments through a memetic algorithm using both matchfmeasure and unanimous improvement ratio. Artif Intell 223:65–81
26. Kuhn HW (1955) The Hungarian method for the assignment problem. Nav Res Logist Q 2(1–2):83–97
27. Acampora Gi, Vitiello A (2020) A study on local search meta-heuristics for ontology alignment. In: Computational intelligence for semantic knowledge management. [S.l.]. Springer, pp 53–70
28. Yager RR (1988) On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans Syst Man Cybern 18(1):183–190
29. Hertling S, Portisch J, Paulheim H (2019) Melt-matching evaluation toolkit. In: International conference on semantic systems. [S.l.]. Springer, pp 231–245
30. Paulheim H (2019) Evaluating ontology matchers on real-world financial services data models
31. Biniz M, El Ayachi R (2018) Optimizing ontology alignments by using neural NSGA-II. J Electron Commerce Organ (JECO) 16(1):29–42
32. Xue X, Ren A. A large scale multi-objective ontology matching framework. In: International conference on intelligent information hiding and multimedia signal processing. [S.l.], 2017. Springer, pp 250–255
33. Tomer SAGI, Avigdor GAL (2018) Non-binary evaluation measures for big data integration. VLDB J 27(1):105–126
34. Xue X, Wang Y, Ren A (2014) Optimizing ontology alignment through memetic algorithm based on partial reference alignment. Expert Syst Appl 41(7):3213–3222
35. Sarasa G, Granados A, Rodriguez FB (2019) Algorithmic clustering based on string compression to extract p300 structure in EEG signals. Comput Methods Programs Biomed 176:225–235
36. Xue X, Tsai PW, Wang J (2017) Using compact memetic algorithm for optimizing ontology alignment. ICIC Express Lett 11(01):53–58
37. Marjit U (2015) Aggregated similarity optimization in ontology alignment through multiobjective particle swarm optimization. Int J Adv Res Comput Commun Eng 4(2):258–263
38. Essayeh A, Abed M (2015) Towards ontology matching based system through terminological, structural and semantic level. Procedia Comput Sci 60:403–412
39. McBride B (2002) A semantic web toolkit. IEEE Internet Comput 6(6):55–59
40. Euzenat J, Shvaiko P (2013) Ontology matching, 2nd edn. Springer, Berlin
41. Banerjee S (2003) Extended gloss overlaps as a measure of semantic relatedness. In: Pedersen, T (ed.) Ijcai. [S.l.: s.n.], vol 3, pp 805–810
42. Winkler WE (1999) The state of record linkage and current research problems. In: CITESEER. Statistical Research Division, US Census Bureau. [S.l.]
43. Damerau FJ (1964) A technique for computer detection and correction of spelling errors. Commun ACM 7(3):171–176
44. Xue X, Chen J (2019) Optimizing ontology alignment through hybrid population-based incremental learning algorithm. Memet Comput 11(2):209–217
45. Xue X, Liu J (2017) A compact hybrid evolutionary algorithm for large scale instance matching in linked open data cloud. Int J Artif Intell Tools 26(04):1750013
46. Schütze H, Manning CD, Raghavan P (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
47. Acampora G, Kaymak U, Loia V, Vitiello A (2013) Applying NSGA-II for solving the ontology alignment problem. In: 2013 IEEE international conference on systems, man, and cybernetics (SMC), [S.l.]. IEEE, pp 1098–1103
48. Xue X, Liu J (2017) Optimizing ontology alignment through compact MOEA/D. Int J Pattern Recognit Artif Intell 31(04):1759004
49. Xue X, Liu J (2017) Collaborative ontology matching based on compact interactive evolutionary algorithm. Knowl Based Syst 137:94–103

50. Xue X, Lu J, Chen J (2019) Using NSGA-III for optimising biomedical ontology alignment. CAAI Trans Intell Technol IET 4(3):135–141
51. Xue X, Chen J, Chen J, Chen D (2018) A hybrid NSGA-II for matching biomedical ontology. In: International conference on intelligent information hiding and multimedia signal processing. [S.l.]. Springer, pp 3–10
52. Xue X, Chen J (2019) Using compact evolutionary tabu search algorithm for matching sensor ontologies. Swarm Evol Comput 48:25–30
53. Xue X, Wang Y (2017) Improving the efficiency of NSGA-II based ontology aligning technology. Data Knowl Eng 108:1–14
54. Zhang F, Guo Z (2019) A novel optimization method for ontology matching based on heuristic population evolution algorithm. Arab J Sci Eng 44(4):3137–3153
55. Forsati R, Shamsfard M (2016) Symbiosis of evolutionary and combinatorial ontology mapping approaches. Inf Sci 342:53–80
56. Xue X, Pan JS (2017) A segment-based approach for large-scale ontology matching. Knowl Inf Syst 52(2):467–484
57. Xue X, Liu J, Tsai PW, Zhan X, Ren A (2015) Optimizing ontology alignment by using compact genetic algorithm. In: 2015 11th international conference on computational intelligence and security (CIS). [S.l.]. IEEE, pp 231–234

Nicolas Ferranti holds a M.Sc. (2020) and a B.Sc. (2017) in Computer Science from the Federal University of Juiz de Fora (UFJF), Brazil. He is currently a Ph.D. student and a teaching and research associate at the Institute for Data, Process and Knowledge Management at Wirtschaftsuniversität Wien (WU Vienna), Austria. His main research interest is on data-centric approaches for information systems. This includes topics in the field of the semantic web, data integration, natural language processing, and knowledge representation.



Jairo Francisco de Souza holds a Ph.D. (2012) in Informatics from the Pontifical Catholic University of Rio de Janeiro, Brazil. Jairo is a full professor in Information Retrieval and Natural Language Processing, member of LApIC research group at Federal University of Juiz de Fora, Brazil. He has worked on several research projects funded by the Brazilian National Research and Educational Network (RNP), Coordination for the Improvement of Higher Education Personnel (CAPES), National Council for Scientific and Technological Development (CNPq), and others. His research interests are knowledge representation, semantic web, information integration, and natural language processing.

Stênio Sã Rosário Furtado Soares from the Fluminense Federal University, Brazil. Graduated in Computer Science from Federal University of Piauí (1999), Master in Applied Computing and Automation from Fluminense Federal University (2004). Has experience in teaching in the field of Computer Science. Professor at the Federal University of Juiz de Fora. Areas of interest: Combinatorial Optimization, Algorithms, Computational Intelligence and Theory of Computation.