IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 22, 2022, accepted March 15, 2022, date of publication April 1, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3164067

Matching Network of Ontologies: A Random Walk and Frequent Itemsets Approach

FABIO SANTOS[1,2], AND CARLOS E. MELLO[1]
[1]Programa de Pós-Graduação em Informática (PPGI), Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro 22290-255, Brazil
[2]School of Informatics, Computing, and Cyber Systems (SICCS), Northern Arizona University, Flagstaff, AZ 86011, USA

Corresponding author: Fabio Santos (fabiomarcos.santos@uniriotec.br)

This work was supported by the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) under Grant E-26/210.231/2021.

ABSTRACT A System of Systems (SoS) is a complex set of IS (Information Systems) created by the aggregation and interconnection of ISs. SoS brings unexpected behavior and functionality during its construction to the benefit of its users and the SoS itself. The integration of SoSs is a matter of time. However, as SoSs can behave as an organization, it may be inappropriate to try to integrate individual IS members separately. On the other hand, manually integrating the SoS as a whole can be unfeasible due to its complexity. If an SoS has ontologies modeling the knowledge, the integration of SoSs can be translated into a problem of a network of ontologies alignment. However, it creates another challenge: computing each possible pair of entities inside each network's ontology can have unfeasible execution time, even using the best matchers available. In this article, we propose to mine the data from the networks using random walks and frequent item sets algorithm and discover relevant nodes elected as candidate entities. Next, the networks are pruned by an algebraic method eliminating identical entities. The relevant nodes are reinserted in the network to avoid losing essential correspondences. After the pre-processing step, data is sent to two matchers to obtain metrics and compare the results with the pairwise brute force approach and previous work. We identified relevant nodes with recall up to 0.75. The results are promising since precision and recall are closer to the force brute, and execution time is shorter, even more, when the size of the networks and the number of ontologies to be compared increases. We validate our approach using ontologies created from the OAEI (Ontology Alignment Evaluation Initiative).

INDEX TERMS System integration, system of systems, ontology matching, network of ontologies.

I. INTRODUCTION

System of Systems (SoS) became a natural evolution of Information Systems (IS). This evolution began with the enhancement of network protocols and the popularization of the Internet, and has been continuing up to these days, in which IS are unable to operate alone. This evolving process has been driven, especially, by user demands for new requirements. Many ecosystems such as e-commerce, m-commerce and social networks are interesting examples of SoS that have been quickly evolving together so that no boundaries among them can be perceived. Many important applications of SoS have been proposed in different domains, such as Smart Cities and Integrated Healthcare, Emergency Response, and Crisis Management Systems [11].

In this context, integration has become a highly important feature of SoS, especially due to requirements such as built-in authentication and cross-platform data sharing solutions in order to connect different resources from different applications. Despite these needs for integration, the IS must remain operating with independence and management capacity. On the other hand, the SoS must support the collaboration among the IS providing interoperability. The SoS works as one entire system, being able to address not only those distinct features held by its compound IS, but also the new ones that emerge from the resulting behavior provided by such a collaboration [29].

For instance, when one company acquires another that operates in the same business domain, usually both have similar IS or SoS. Therefore, in order to keep the operations running smoothly and steady, one should conduct an integration process between these SoS. However, this can be very challenging due to the multiple possibilities of knowledge and

The associate editor coordinating the review of this manuscript and approving it for publication was Francisco J. Garcia-Penalvo.

concept representation shared by both SoS. Then, one may appeal to ontologies or network of ontologies that represent pieces of knowledge required for the integration of the SoS. In this way, the integration process can begin with the matching of these networks of ontologies as it may offer a common ground for the posterior SoS integration.

Accordingly, the goal is to first integrate the pieces of knowledge throughout the concepts related to each associate SoS, and later utilize the resulting mapping to conduct the SoS integration itself. The matching task of ontologies and network of ontologies essentially consists of finding equivalent concepts, a.k.a. correspondences or matches.

This task of matching networks of ontologies can be handle as an *ontology matching* problem, which basically consists of finding correspondences between two or more ontologies. However, to extend it for network of ontologies one should deal with computational issues due to the large number of possible correspondences to be considered between two or more networks. Hence, we may address this task as a different one, in which one should overcome the exhaustive computations by taking advantage of underlying structural properties held by the networks of ontologies [10], [39]–[41].

We conducted a case study [47] using the conference domain of ontologies project [2] as our case study. OAEI is an initiative to help improve the work on ontology alignment/matching, provide communication between developers, promote conferences, assess the strengths and weaknesses of alignment/matching systems, compare the performance of techniques, and improve evaluation techniques. OAEI provides datasets[1] of ontologies with reference alignments to create an environment that compares the research evolving ontology matching. Since 2004 the OAEI has evaluated tracks oriented to different ontologies, sizes, types of alignment, and modalities. The last results had 13 evaluations. The conference domain had ten competitors in 2020 and is one of the essential datasets being refined since 2006. We chose the conference domain dataset for our case study because of the reliability of the reference alignments and as a way to compare our research results with many results available from the OAEI competitions.

A. PROBLEM DEFINITION

This study addresses the characteristics of ontology matching in networks with enormous numbers of entities by trying to avoid computing all possible matchings.

Some problems have the characteristics of being too complex to be processed by computers and, consequently, to provide an analytical solution. A possible solution to these problems is to better represent the occurrence of the problem by randomly sampling the possible results. The intuition behind this research comes from the idea that the ontology structure contains information about the relevance of entities. The relative importance of a concept in one ontology might be the same as its counterpart in the other ontology. Therefore, a concept more central in one ontology tends to also appear more centrally in the other ontology. On the other hand, a more peripheral concept in one ontology also tends to be more peripheral in another ontology. Also, the corresponding concepts are structurally represented in two ontologies in a similar way, and therefore will be present with similar relative frequencies within samples generated from a sampling process that respects the distribution of the structure between the concepts. As the research aims to avoid computing all similarity metrics between each pair of concepts (pairwise approach), we can sample the most relevant concepts by randomly selecting a subset of the ontology.

The method goal is to reduce the comparison effort without pruning relevant entities, therefore reducing the accuracy. Our proposed method does not compete with the large-scale matching strategies, which are orthogonal. While large-scale techniques try to identify similar modules to restrict the comparison scope using a heuristic, our approach combines the identification of regions with the same entities to prune them and avoid the comparison and a stochastic search method to keep the relevant concepts. They are complementary strategies: we can use our method to prune the same entities in both networks, reinsert the relevant and finally send them to a large-scale matcher.

The research questions we want to answer are:

RQ1: To what extent the method can identify the relevant entities? To answer RQ1, we employed a random walk and frequent itemsets approach to identify those relevant entities. We also explored the influence of configurations (i.e., quartile limits to keep the path visited and a semantic threshold to keep a mapping suggesting). Overall, we found that pre-processing the ontology can discover relevant nodes with an average recall of 75%

RQ2: To what extent do the relevant entities improve the matching of the network of ontologies? To answer RQ2, we ran experiments matching networks and comparing with the study from [41]. We also compared it with two classic matchers. The metrics obtained are compatible with the classic matchers and reduce processing time in large networks. The results showed they still cannot compare complex structures with many sources of reference alignments.

We evaluated the proposed approach in a preliminary experiment using an OAEI dataset.

The contribution of this study is: that we develop a method able to match an extensive network of ontologies in better execution time than the pairwise approach with similar metrics. We created open-source code available to researchers addressing large-scale and network matching.

II. BACKGROUND

A. ONTOLOGY

Ontology is an explicit specification of a conceptualization [12] and a set of representational primitives with which to model a domain of knowledge or discourse [13]. The Computer Science area had realized the importance of the

[1] http://oaei.ontologymatching.org/2021/

ontologies to construct a foundation where Information Systems may be developed with meaning stability. Once a domain or task has an ontology to refer to, all the technicians may precisely understand the kinds involved. Guarino and Giaretta called that as "ontological engineering" [14].

After understanding why the ontologies may be constructed, they have been built for many domains. The myriad of ontologies being built leaded to a multi definition of concepts that caused problems to solve queries and difficulties to integrate systems where ontologies are in background to support the knowledge model. Thus, to integrate IS, ontologies need to be aligned, enabling a translation of concepts. Align the concepts or entities becomes crucial in situations such as systems integration within the company or outside, such as in electronic commerce, when a company needs to map the concepts between its systems and systems belonging to partners, suppliers, and customers [26].

B. ONTOLOGY MATCHING

The process of aligning ontologies, finding and describing their matches is called ontology matching [44]. Thus, an alignment is a set of correspondences that express the translation semantics between concepts belonging to different ontologies. Given two ontologies, correspondence is a 5-uple: $< id; e_1; e_2; r; c >$, where the first element, id represents a unique identifier for the correspondence, e_n are classes or properties (entities in general), r is the relationship, considering $r \in \{=, \sqsubseteq, \sqsupseteq, \perp\}$. equivalence, subsume (more and less general) and disjointness, and the confidence value in the interval from 0 to 1 [44]. The relationship implies the alignments are subject to structural constraints, which must be obeyed to maintain consistency [25].

C. PAIRWISE MATCHING

Given a set of ontologies, if the pairwise matching is applied to all ontologies from this set, it will sequentially compute the alignment of each pair of ontologies from the set. For example, given a set of ontologies $\Gamma =< \Omega, \Lambda >$, in which $\Omega = \{O_1, O_2, O_3\}$ of Figure 1, the pairwise matching of all ontologies in the set is obtained by computing $(((O_1 \times O_2) \cup (O_1 \times O_3)) \cup (O_2 \times O_3))$. Thus, the pairwise matching approach computes one pair each time. The final result may have duplicate alignments.

D. HOLISTIC MATCHING

Given a set of ontologies, if the holistic matching is applied to all ontologies from this set, it will compute at once the alignment of all ontologies from the set. For example, given a set of ontologies $\Gamma =< \Omega, \Lambda >$, in which $\Omega = \{O_1, O_2, O_3\}$ of Figure 1, the holistic matching is obtained by computing $(O_1 \times O_2 \cup O_1 \times O_3 \cup O_2 \times O_3)$. Thus, the holistic matching approach computes all matchings between all ontologies inside this set and merge the results deleting duplicate alignments.

E. NETWORK MATCHING

A network of ontologies is a set of two or more ontologies aligned. We define a network of ontologies $\Gamma =< \Omega, \Lambda >$ a finite set Ω of ontologies and a finite set of alignments between these ontologies and $\Lambda(O, O')$ represents the set of alignments in Λ between O and O' [10]. Given a set of two or more networks of ontologies $\Psi = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$, the network matching problem searches for a final network of ontologies Γ_f resulting from the alignments of the networks in Ψ. For instance, Figure 1 depicts two networks of ontologies (Γ and Γ', each one with 3 ontologies ($\Gamma = \{o_1, o_2, o_3\}$ and $\Gamma' = \{o_{1'}, o_{2'}, o_{3'}\}$, describing two Systems-of-Systems. The goal is to match these two networks, finding all the alignments between them. The set of alignments between the networks (or inter network alignments) are: $\Lambda = \{A_{2,2'}, A_{3,2'}, A_{3,1'}\}$.

F. LARGE-SCALE ONTOLOGY MATCHING

Large-scale matching and efficient matching techniques were defined as challenges by [44]. In this context, the alignment of networks can be considered a variation of large-scale matching with efficiency. In general, these challenges seek to improve the alignments of large ontologies while maintaining good metrics and scalable computational complexity. Pairwise and holistic approaches, where each entity of the source ontology we need to compare with all possible entities of the destination ontology usually make the approach not scalable with the alignment of large ontologies. The main issues faced are: high memory consumption, increasing complexity of the alignment process, and increasing time taken to achieve alignment [34].

The techniques to address both efficiency and scalability can be summarized as: reduce the search space and parallel the matching [37].

To reduce the ontology search space, ontology partitioning techniques can be used. Partitioning aims to decrease processing time while allowing space complexity to be reduced. In a pairwise approach, the search space is $O(n^2)$, considering n the number of entities to be compared. Partitioning aims to create regions that limit the scope of comparisons. Thus, partitioning can reduce this space down to $O(n^2/k)$, where k is equal to the number of partitions created. Partitioning also influences the processing time. Assuming that the computation time of the similarity metrics for each entity pair is t, a naive pairwise approach would use $O(n^2) * t$ time units to get all the metrics. By partitioning the ontologies, we can reduce this time to $O(n^2/k) * t$. However, partitioning introduces a risk: if the partitioning algorithm creates poor partitions, it will impact the final metrics (precision, recall, and F-measure) and limit the comparison of entities between similar partitions.

Two common approaches are those based on graphs and those based on ontology logic. Therefore, the graph approach uses only the ontology structure and is usually more scalable but ignores part of the semantics that extrapolates the

TABLE 1. Weakness of network alignment approaches.

factor	pairwise	holistic	subinternm
Merge the results	No	**Yes**	**Yes**
Prune identical entities	No	No	**Yes**
Number of comparisons	Ω X Ω'	Ω X Ω'	**<** Ω **X** Ω'
Change the structure	**No**	**No**	Yes

information obtained from the structure. The logic-based technique allows the use of description logic and reasoning with axioms, better capturing the semantics described by the ontology, and therefore, as it depends on the reasoning processing, it is less scalable.

G. NETWORK OF ONTOLOGIES AND SYSTEM OF SYSTEMS

Networks of ontologies can be used to support a System of Systems (SoS). SoS is defined as a set of independent systems, providing functionalities derived from the interoperability between them [5].

An extensive network of ontologies can contain several of the same entities. Each ontology in the network describes its domain or complements the knowledge of another ontology in the same domain. When we integrate systems from two companies, we may have networks with ontologies that describe the same or similar domains. These domains can be supported by the same ontologies or by different ontologies.

When we think of an environment with multiple IS and therefore multiple ontologies, the alignment process can be highly complex and inefficient if naively tried comparing all possible existing pairs of entities of all ontologies inside two networks of ontologies.

Holistic matching addresses the duplication in results when matching many different networks. However, both pairwise and holistic matching are unable to address the exponential increase of the number of comparisons (table 1).

SubInterNM [41] addresses the case when ontologies share same entities and prune them. Indeed, it avoids the Cartesian product. However, it creates a side effect: when the network of ontologies are converted to graphs the entities identified as identical, based on the URI, are pruned from the graphs. Thus, they are excluded from the final alignment result and the structural metrics, gathered from the graphs, might be harmed. In figure 1 the entire ontology o_2 does not need to be compared with the identical o'_2 since they shared the same entities. The same occurs for entities a_1 and b_1 from ontology o_1 and a'_1 and b'_1 from ontology o'_1 and a_3, b_3, c_3, d_3, e_3 from ontology o_3 and their counterparts in ontology o'_3

H. MARKOV CHAINS

Markov chains are stochastic problems that can be used to represent complex systems impacted by aleatory changes. Markov property states that the current state depends solely on the state before. Mathematically one can define it as:

$LetV_n = V_1, \ldots V_n$ random variables that satisfy the conditional dependency property. Suppose $s_i, \ldots s_n$to be all possible states random variables can assume. So, according Markov property the probability of a random variable $P(V_n = s_n|V_{n-1} = s_{n-1}) = (P(V_n = s_n|V_{n-1} = s_{n-1}, V_{n-2} = s_{n-1}, \ldots, V_0 = s_0)$.

A transition matrix T for a Markov chain V in time t is a matrix that persists values from transition probabilities among their states. Given a matrix's rows and columns by the state space, the element of the matrix Ti,j is given by

$T_{i,j} = P(V_n = s_n|V_{n-1} = s_{n-1})$. It is important to highlight that the sum of the matrix lines $= 1$ and it is a probability vector.

Using these concepts in ontologies, the transition matrix can be represented by the adjacent matrix used in graphs to determine the adjacent nodes. Assuming that a node has four child nodes and considering that the probability of a transition is equal for each node, the transition matrix for this node will be [0.2,0.2,0.2,0.2,0.2] where the first 0.2 means the probability of the next stage is the same node, while all other 0.2 values are the probabilities of a child node being selected for the next state.

I. RANDOM WALK

A random walk (RW) is a finite Markov chain and has a finite set of states. A random walk starts at some state. At a given time step, if it is in state x, the next state y is selected randomly with probability p_{xy}. A node of the random walk after sufficiently many steps is therefore essentially uniformly distributed [27].

They have a direct relation with kinds of Markov chain depending on the underlying data structure will be explored. Indeed, is a model of a stochastic process that serves to solve a wide variety of problems and applied and a vast range of domains such: as probability theory, computer science, statistical physics, operations research, and others [30]. Some problems list include financial markets, social network affinities, and ranking systems [30].

RW and graphs data structure can be combined in a computational environment. Both have been used to model problems in several domains. Both can be customized to model complex systems behavior and the corresponding data model [27]. The authors in [30] discuss many RW types and applications according to the type of graph representing the networks. They also stated the RW is good to "uncover various types of structural properties of networks, identifying "central" nodes, edges, or other substructures in networks."

Let $G = (V, E)$ be a connected graph with n nodes and m edges. Suppose in time t a movement is made from node v_t to v_{t+1}. Consider d_t the degree of node v_t. The probability of reach $v_{t+1} = \frac{1}{d_t}$, if G is a graph and $\frac{1}{d_t+1}$ if G is a digraph, if $t, t+1 \in E$ or 0 otherwise.

The sequence of steps can be seen as a Markov chain where the distribution probability of being at node t is p_t.

The transition matrix M contains all $P_{t,t+1}$ representing the probability to reach node $t+1$.

Since ontologies can be represented as a graph it is possible to use an RW to go through the ontology structure.

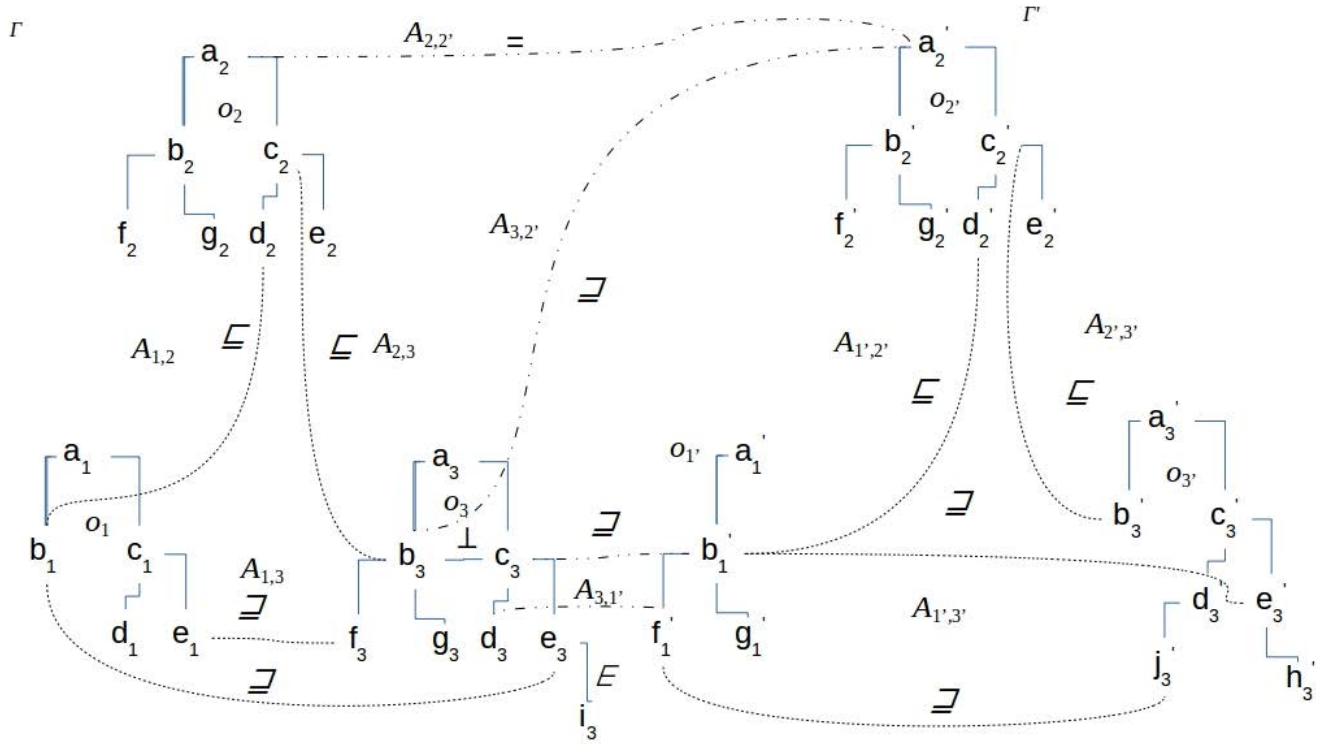

FIGURE 1. Matching network of ontologies example adapted from [10].

RW algorithms also can deal with specific situations and problems intrinsic in networks structure. In fact, variations on the networks created a fertile terrain to enrich the RW research. Moreover, that symbiotic environment encourages both areas to grow together. The work of [30] lists many of them: RW variations, types of networks, and RW algorithms providing a brief survey. Although the author did not mention the network of ontologies domain, it is possible to represent the network structure with a graph as used in [40]. Therefore, RW may be used to address the network alignment problem as well.

Consider a RW movement inside a structure that represent an ontology or a network, and the transitional probability matrix M^t. Suppose in such moment a movement will be done from position v_t to position v_{t+1}, and probability P_{ij}. So each element i, j from M^t is: $M^t_{i,j} \frac{P_{ij}}{d_i}$

Markov property hold for network RW, so the following remains true: $\sum_{j=1}^{N} M^t_{ij} = 1$

So, it is possible to gather all the visits running many random walks through the structure and create some expectations about the more visited nodes and edges, and also these information can reveal some tips about the position of the nodes and edges on the whole structure. In fact, because of the probability of the transitions, some nodes can be more visited than others. It might create a suggestion of the node relevancy. The work of [45] discusses the node importance based on this centrality and the spectrum of the adjacency matrix.

Following the random web surfer model and the page rank algorithm [16], [35], the rank of a such node can be calculated based on the degree of connectivity. The higher are the number of incoming edges the more importance may have a node. Also, the higher is the position in the graph hierarchy more the node importance. Since each RW execution is starting in a random node the hierarchy position, it is not being used in this study.

The formalization of this problem defines the concept of neighborhood $n(i)$ which is all indexes j pointed by the node i. So, $n(i) = i : (j, i) \in G(V, E)$. Suppose $N(i)$ as the number of i neighborhood and $n^{-1}(i)$ the neighborhood that points to i.

The rank of a node is given using a notion of visit time. Suppose a node i is visited at the time $t = 0$ and at time $t = 1$ one of its neighbors is visited. The time spent at the node defines its rank. If we define a function:

$$I(W_n = i) = \begin{cases} 1 & \text{walker } W \text{ visits node } i \text{ at time t,} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

So, the rank r_1 of the node i is:

$$r_1 = \lim_{t \to \infty} \frac{1}{n} \sum_{m=1}^{n} I(W_n = i) \quad (2)$$

The method was implemented using the same probability for each node in each transition of the walker $W = \frac{P_{ij}}{d_i} = \frac{1}{d_i}$ in function of degrees of the node i or $\frac{1}{N_i}$.

Then the method performs the RW a large number of times and keeps track of the visits. The rank (r_1) is approximate:

$$r_1 \approx \frac{1}{n} \sum_{m=1}^{n} I(W_n = i) \tag{3}$$

This study proposes a variation of the page rank, random web surfer algorithm, to rank the nodes [16], [35]. Count each node's number of visits, analyze the paths statistically, and the patterns gathered, using random walks and frequent itemsets to determine its relative importance in the graph structure. As the graph structure represents an ontology, considering the data from the paths and the patterns, a highly visited entity can be selected as a candidate for reference alignment. Information about the paths where the entity was visited will be used together to reinforce the choice of entities.

III. RELATED WORK

The ontology matching problem has been researched for a decade. However, despite the evolution in the field, to the extent of our knowledge, matchers still work pairwise, i.e., computing the alignments for each pair of ontology. Thus, there are few studies dealing with the area of ontology networks and, consequently, the problem of ontology network matching, although there are open challenges to deal with [39].

We have found the applications that usually need some matching of networks were: biomedicine, biological networks, link prediction, ontology matching, multi-layer networks, and social networks.

Thus, we looked for similar domains where matching among networks and the complexity of structures have been applied, including the large-scale ontology matching.

A. NETWORK MATCHING

We identified approaches and clustered them into three domains or groups. The domains we found were: AI, databases, and network science. The network science group transforms the problem in a graph representation and try to employ the best ways to solve the graph isomorphism problem or went beyond, addressing a multiple network problem, i.e., not limited to only two networks, and try to predict links to avoid the massive computation of all possibilities.

Prediction is a way to avoid massive computation of all possibilities. Sometimes it is needed because there is little or no information about the nodes. Indeed, it also may be caused by security and privacy situations or because the network to align is a very new structure, e.g., a new social network. The papers from the AI group are concerned with training models, creating neural networks to refine the algorithms to obtain gains in near future alignments, or formally modeling the structures to define possible operations over networks better.

Finally, we have a small group of database domain articles that deal with distribution, algebra operations, and modular structures to reduce the final computation The main goal is ''divide to conquer'' to transform the problem in smaller pieces and, thus, deal with large and intricate structures.

Prime Align evolved the PageRank algorithm balancing a weight representation of the edges and the topology using a random walk and Markov chain [22]. The work from [7] predicts links between networks after determining some consistencies between them, using the fact that two very different domains (social networks and PPI networks in the study), might keep some degree of consistencies. From [46] the random walk algorithm was used after the generation of an associated graph, which has a system of ranking of affinity. Random walk highlighted the probably most crucial nodes in the graph structure while submerged the others, to select the final set of candidate entities. The work of [19] uses a random walk with a state to define, and whether or not, the next step will occur in on, different networks using the topological similarity computing a transitional probability matrix. The matrix is optimized, and a greedy algorithm outcomes the multiple network alignments. The approach in [31] uses a clustering strategy to reduce the processing step in alignments by dividing and distributing processing parts using a distributed solution called spark.

Our study mixed up some strategies from [46] and the approach presented by [41] trying to refine the preprocessing step with three phases.

B. LARGE-SCALE MATCHING

Previous studies tried to address the complexity of large-scale matching by using diverse strategies. We focus on the graph-based ones since our method explores them.

The work from [42] tries to define a dependency function to create modules who capture similar nodes. Leftovers are connected with the best modules. [33] uses a query-based approach similar to database views to define similar regions starting from some concepts. [24] proposes a taxonomy-based partitioning for gene ontologies which can split partitions by terms. The approach uses up and down propagation to define the boundaries of each partition. The study from [15], computes coupling and cohesion of entities in each proposed cluster (or partition) by computing the value of the entities connected.

Similarly [18] uses the notion of cohesion and coupling to create blocks. However, after defining them, the blocks are connected by anchors avoiding the Cartesian product. The more anchors found between blocks, the more connected they must be matched. Anchor-flood is presented in [43] exploring anchors to create segments using a greedy algorithm. Once all alignments, parents, and descendants are explored, they have the overall alignment. The segments are eventually created after and not before the alignment.

Finally, LogMap [21] uses the ISUB algorithm to expand anchors using a string-based approach. The context grows, including new neighbors without comparing them more than once and only over a predefined threshold. The anchors are chosen using the locality principle: if the hierarchy neighbors of the classes in an anchor match with low confidence, then the anchor is abandoned. Thus, avoiding the Cartesian product.

The previous studies that deal with large ontologies usually try to modularize them in order to avoid explicit comparison to each entity [34]. However, large-scale related work does not consider a previous network structure, and the modularization process clusters regions of the ontology to reduce the problem size and, indeed, does not avoid computing each pair inside the regions. Besides that, an ontology network may be composed of large ontologies that bring the problem to a more significant number of entities comparisons. Thus, modularization and our proposed method may be complementary.

The proposed approach aims to reduce search space while pursuing alignment efficiency. This is possible by recovering possible alignments lost after pruning the graph of networks. The SubInterNM approach creates partitions not to obtain segmentation for creating clusters or modules that reduce the scope of entity comparison. It seeks to discover partitions of equal content, removing them and thus obtaining the reduction of search space. Our method works at this moment. As SubInterNM partially destroys the graphs by pruning similar entities, we discover possible relevant entities by mining potentially crucial nodes, returning them to the graph, and thus recovering the lost structure.

We identified many places where parallelization could assist reduce the time complexity. Comparing our method with the taxonomy of large-scale matching approaches [34], since the reference classify modules when they are created by reasoning, we can classify our method as a no modular extraction and no complete partitioning with possible parallelization using graph-based search space reduction and it is scalable. Indeed, it is a partial partitioning or a structure-based modular extraction

The novelty of this study lies in the proposed method to address large-scale matching evolving networks of ontologies by taking advantage of modeling ontologies as Markov chains from which observations are drawn to be statistically evaluated by data mining techniques to recommend pair matches eventually.

IV. METHOD

This study comprises three phases, as summarized in Figure 2: gathering information from the structures through a random walk, selecting the most relevant entities, and optimizing the network to be matched. To foster reproducibility, we provide a publicly available dataset[2] containing the raw data, the random walker used in phase 1, the Jupyter notebook scripts used in phase 2, and the network optimizer used in phase 3.

A. PHASE 1 - GATHERING INFORMATION FROM THE STRUCTURES

We converted the OWL structures from each one of the dataset ontologies into graphs. Table 2 shows the graph representation created for each ontology in the dataset.

[2]https://zenodo.org/record/5573204

TABLE 2. Ontologies graphs.

Ontology	Nodes	Height	Bags	itemsets
conference	60	7	1597	492
confof	39	3	799	68
edas	104	4	2245	263
ekaw	74	6	875	1141
iasted	141	5	4091	553
sigkdd	50	4	1016	137

After, the random walk algorithm ran ten times the number of nodes in each ontology to retrieve data from the nodes visited. This information includes the frequency, paths, and distribution of each random walk, including the mean, median, quartiles, max, and min values.

We filtered out the paths during the data collection using a sliding window algorithm limiting the data gathered from the random walks. The sliding windows were set up to start in (1,1) (size of window, size of shift) until the window's maximum size could retrieve a valid path. We also filtered out paths from random walks based on the number of visits considering the quartiles one, two, and three from the statistical distribution for each possible sliding window configuration. We kept only paths that had at least one node inside the quartile. Thus, we generated the number of possible random walks x 3 output files for each ontology. For example, one ontology with valid random walks starting from (1,1) until (5,5) (i.e: (1,1), (2,1),(2,2), (3,1), (3,2), (3,3),(4,1), (4,2), (4,3),(4,4), (5,1), (5,2), (5,3), (5,4), (5,5) X 3) generated 45 different output files considering the three defined quartiles.

The main output files were the visit file, summarizing each node with the total number of visits and the paths where it was present, and the binary file containing each valid path 1 or 0 indicating whether or not a specific node was visited.

Figure 3 shows the random walk sliding window algorithm gathering data from the sigkdd ontology and creating the main output files: visits and binary. This experiment focused only on the Class node types for feasibility when manually analyzing the results from each step in each phase.

B. PHASE 2 - SELECTING THE MOST RELEVANT ENTITIES

Once we have information about the nodes with potential more relevancy in the graph structure representation of the ontology, we need to discover the best setup to maximize the metrics and persist the better overall setup for each ontology and the entire domain.

The data mined by the random walker made a large amount of data available for analysis, specifically in our case, the many sequences of paths visited within the graph structure, in the form of unstructured text, for each different configuration to identify possible relevant nodes. The complexity of unstructured text has created new challenges in data analysis [6], and data aggregation algorithms may help to identify relevant data. In order to evaluate all possible

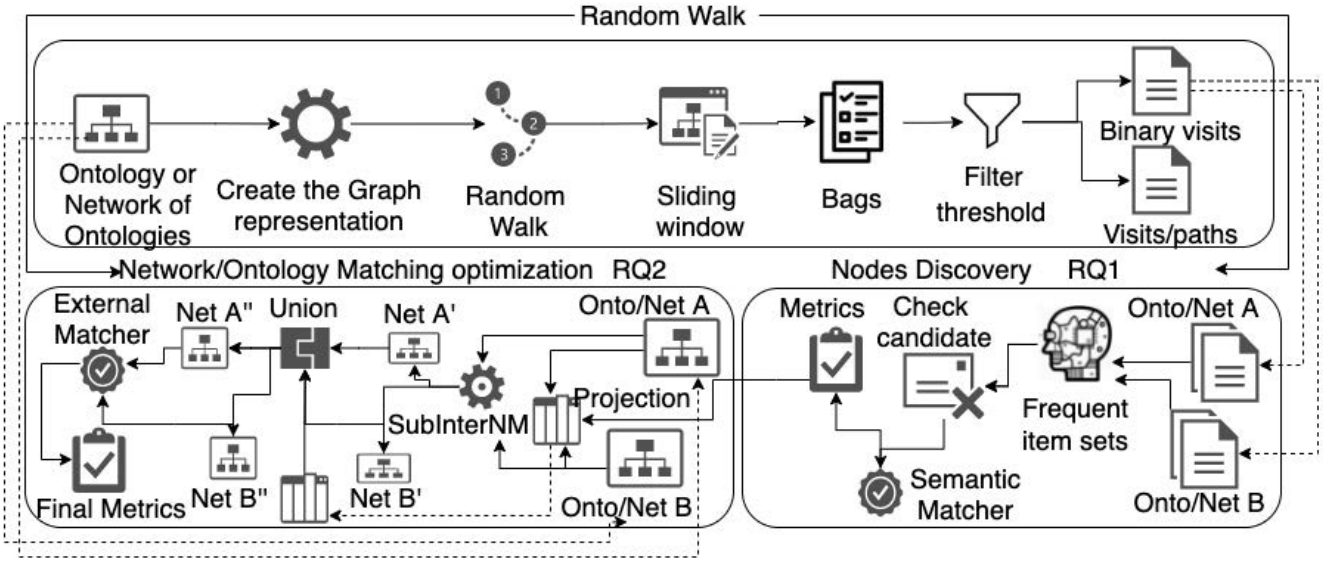

FIGURE 2. The Framework of Research Design.

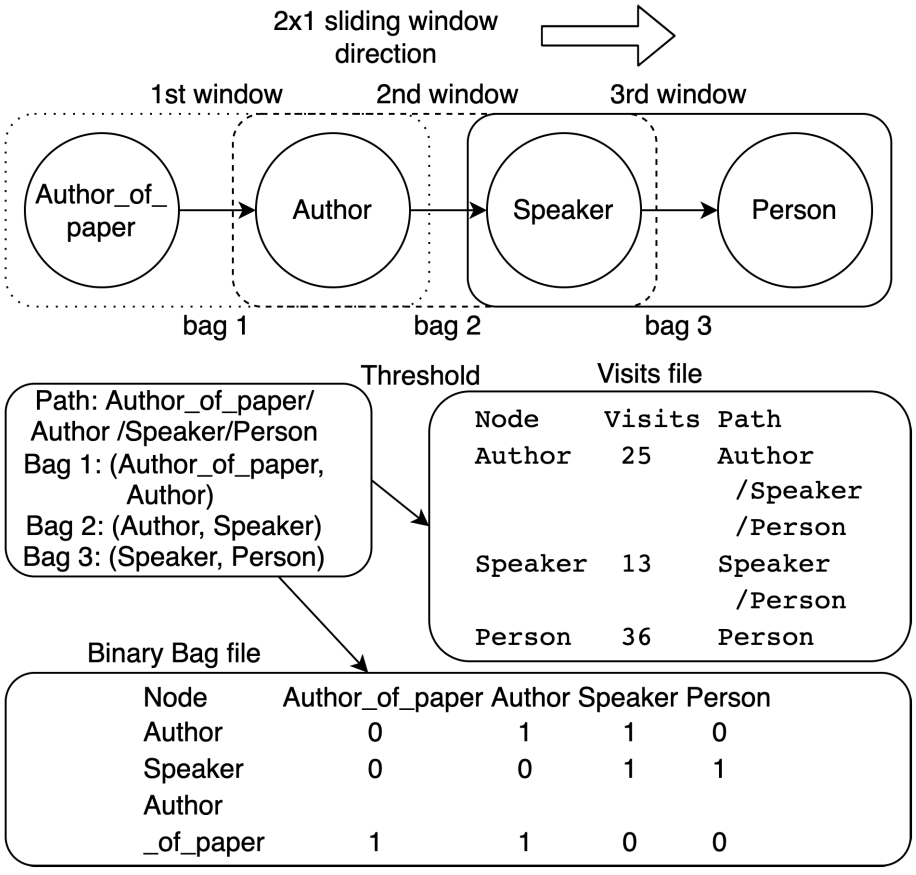

FIGURE 3. Phase 1 - sigkdd ontology example.

configurations, we aggregated the data, identified patterns, and applied natural language processing to evaluate the possible candidate hits by comparing the chosen nodes to the reference alignment.

1) FREQUENT ITEMSET EVALUATION

To select the most relevant nodes to the next phase, we ran a frequent itemsets algorithm using all the binary outputs produced in phase 1.

We used the Python Mlxtend package [1] to run the apriori algorithm to obtain the frequent itemsets ordered by the support. The apriori algorithm is simpler than its counterparts, Eclat and FP-Growth. However it can overload the memory when bags have more than 60 itens [17]. It did not happen in our dataset. We build an evaluation experiment for each pair of ontologies present in the OEAI dataset with a reference alignment.

Each ontology was evaluated separately and incrementally, including the next bag sorted by the support. After the bag was collected, we verified whether or not each bag produced by the apriori algorithm was unique to avoid comparing the same bags multiple times. Each bag is inserted into a collection that does not allow duplicates. We called this the candidate check.

Once the candidacy was approved, we individually compared the bag with the reference alignment from both ontologies (each side). Then we retrieved the recall, precision, and F-measure evolution for each side separately. For example, a pair of ontologies that ran the random walk from (1,1) to (5,5) and three quartiles have 45 (RW output files) x 2 (ontologies) x num of bags side evaluations.

From each side evaluation and the final evaluation, we gathered the graph evolution of the recall, precision, and F-measure to track the progress of the metrics after each bag was incorporated into the process.

We generated a data set for each sliding window + quartile + semantic threshold to compare the results and select the best setup for each pair of ontologies.

Finally, we compared all the results from each pair of ontologies to define the best overall setup to feed the network matching optimizer in the next phase.

2) SEMANTIC THRESHOLD

Next, we to discard 'noisy' candidates we defined three levels of semantic thresholds: 0.7, 0.8, and 0.9 mimicking the range used by LogMap and Alin. LogMap uses for a default expansion a threshold of 0.70 and mapping threshold of 0.95 [21]. While in Alin, using the Conference dataset, the similarity value was defined as 0.9 [9]. We applied the Spacy NLP Python package to compare each bag of frequent itemsets between the ontologies. The Spacy package [3] was trained using the model (*en_core_web_lg*) that is the largest English model of spaCy with a size 788 MB. Then we combined each possible pair of selected tokens from the unique bags created for each ontology and compared them with the complete reference alignment.

The final evaluation considered the semantic threshold defined above to keep only pairs beyond the limit defined (0.7, 0.8, or 0.9).

Figure 4 shows an example of the phase 2.

3) DATA EVALUATION

To evaluate the classifiers, we employed the following metrics:

- **Precision** measures the proportion between the number of correctly predicted labels and the total number of predicted labels.
- **Recall** corresponds to the percentage of correctly predicted labels among all truly relevant labels.
- **F-measure** calculates the harmonic mean of precision and recall. F-measure is a weighted measure of how many relevant labels are predicted and how many are relevant.

4) DATA ANALYSIS

We used the evaluation as mentioned earlier metrics to conduct the data analysis. We used the Mann-Whitney U test to compare the metrics from the outcome of the frequent itemsets, followed by Cliff's delta effect size test. The Cliff's delta magnitude was assessed using the thresholds provided by [38], i.e. d <0.147 ''negligible,'' d <0.33 ''small,'' d <0.474 ''medium,'' otherwise ''large.''

5) DATASET ANALYSIS

Data mining techniques and machine learning are frankly employed in data analytics applications. Frequent itemsets mining is widely used among the unsupervised techniques to find information about association rules, correlations, and dependencies hidden in datasets. A frequent itemset is a form of a combination of all items in transaction data that is interconnected, and the match can be found using several data mining techniques, one of which is the apriori algorithm [4], [32]. One advantage of the frequent itemsets is the flexibility to be employed with many domains due to the simplicity of the required dataset format. The dataset consists of the visited nodes produced by the random walk algorithm, where each row has columns for all the node entities marked with ''1'' if it was visited in that random walk or ''0'' if it was not.

C. PHASE 3 - MATCHING NETWORK OF ONTOLOGIES

Entering phase 3, we need to use the information discovered in phases 1 and 2 in the future alignment of networks of ontologies. We could analyze and decide the best setup for this case study using different setups. We claim the final set of nodes suggests the structurally more relevant candidates. However, the structural information should not retrieve all relevant nodes to the network match. Thus, increasing the recall, including more nodes, and discarding them only in the final matcher computation, seems more crucial.

1) EXTRACTING THE RELEVANT NODES

Using the best setup outcome from the frequent itemsets, after the data analysis step, we used a projection algorithm defined in [8] to extract the nodes from the network of ontologies. Those nodes will be kept safe for future use. We used the Ontology Manager Tab [28] and the SubInterNM (defined in [40]) modified with the projection operation.

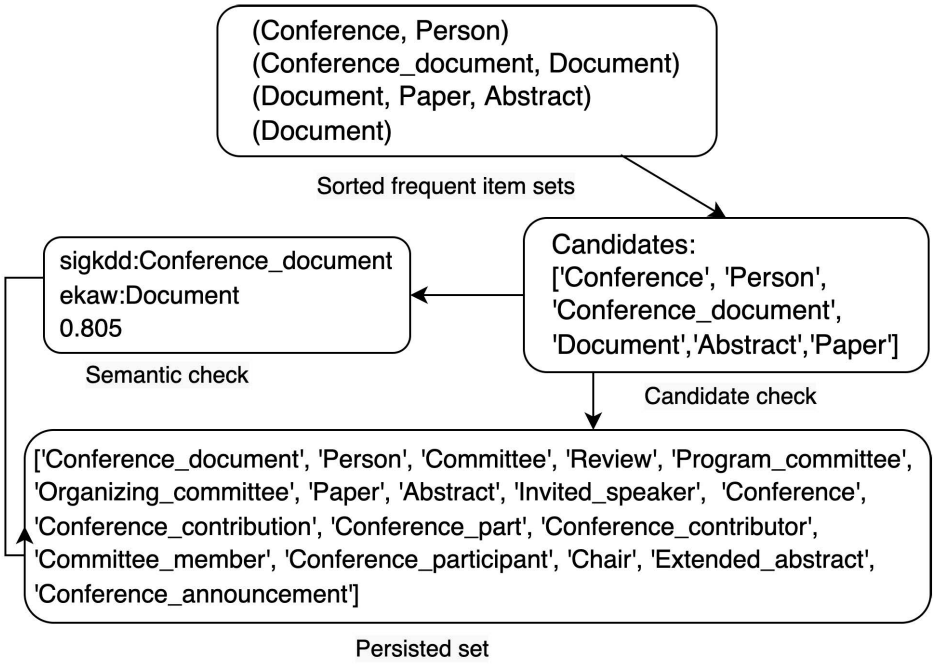

FIGURE 4. Phase 2 - sigkdd ontology example.

2) OPTIMIZING THE MATCHING

Using the graph representing the network structure, we employed the SubInterNM approach preprocessing the networks, pruning redundant entities found in both networks, reducing the final matcher's effort. Indeed, it employs a set of algebraic operations (union, intersection, and difference) to the graph representation of the ontologies. The results discussed in [41] reduced the number of comparisons, avoiding the complete Cartesian product to verify all possible entity matches. Also, the study mentioned above showed the reduced overall time processing and more balanced metrics compared with traditional matchers used alone to deal with networks.

In complement with the outcome from the SubInterNM, we inserted the persisted nodes retrieved from the projection operation back to the graph, using the union operation, sending them back to their respective networks and, consequently, restoring some relevant nodes that the SubInterNM approach has pruned.

To evaluate the experiment, we create networks using the conference domain ontologies. The choice of that domain to compose the networks, despite not reflecting an ideal situation in the real world, was because of the reliability of the reference alignments available to evaluate the results considering many different possibilities. OAEI provides 15 different reference alignments for combinations using seven ontologies (conference, cmt, confof, edas, ekaw, sigkdd and iasted). We excluded the cmt ontology from the experiment due to errors processing the RDF file.

We evaluated the following networks: (table 3).

TABLE 3. Phase 3 experiments.

Experiment	Network 1	Network 2
2x2	sigkdd, confof	conference, confof
4x4	sigkdd, confof, ekaw, edas	conference, confof, ekaw, edas
5x5	sigkdd, confof, ekaw, edas, iasted	conference, confof, ekaw, edas, iasted
5x1	sigkdd, confof, ekaw, edas, iasted	conference
5x2	sigkdd, confof, ekaw, edas, iasted	conference, confof
5x3	sigkdd, confof, ekaw, edas, iasted	conference, confof, ekaw

Each experiment operation was defined with a string to help the execution. For instance, the 2×2 experiment ran the following projections:

- confof_P_conferenceXconfof
- confof_P_confofXsigkdd

The first item means using the confof ontology and running a (P)rojection using frequent itemsets obtained in phase 2 from the combination of ontologies conference and confof. The second line orients to use the confof ontology and run a projection using the frequent itemsets from confof and sigkdd (confofXsigkdd).

The union operation followed a similar notation:

- net1_220_U_confof_P_conferenceXconfof

Thus, this line above represents the sequenced operation: using the first network $\Omega = \{\text{sigkdd, confof}\}$ in experiment 2×2 using the default configuration in SubInterNM (net1220), make a (U)nion with the result of the (P)rojection

from confof ontology and an outcome from the frequent itemsets that evaluated the conference and confof ontology.

The notation aims to automatize the entire experiment connecting all the phases in a row in the future. Table 11 summarizes the operations in some experiments.

Finally, we projected the final union results with all ontologies fragments, returning to the pairwise problem. This step aims to verify to what extent the results increased the metrics obtained in [41], without the problems caused by the limitations of the actual matchers to identify alignments from more than two ontologies.

Section V will discuss which frequent itemsets results were used.

3) FINAL MATCHING

Following the optimizing step, we sent the results to two external matchers: Alin and LogMap [20]. These matchers were selected because they represent some of the best results in the OAEI campaign, considering the conference domain. In addition, Alin can compare multiple ontologies (but not networks), and LogMap has excellent results for large datasets, which is the situation we face in our experiment comparing networks of ontologies.

4) RESULT ANALYSIS

To the extent of our knowledge, there is no network ontology matcher to compare our results. However, we created a baseline selecting random entities from the ontologies to compare with the Random Walk and the Relevant Nodes Discovery results. Also, the final results can be sent to the two matchers selected, using the outcome after the union operation and after the fragment projections. Indeed, this addresses the problem that classic matchers do not usually handle more than two ontologies simultaneously as input.

5) EXPERIMENT SETUP

Phase 1 ran the random walk using as input the ontologies from the Conference domain available in OAEI. We use all possible sliding window and offset setups limited by the height of the graph representation of each ontology. The visits files were limited by the quartile threshold parameter responsible for deleting paths where at least one of the entities visited in each path is under the defined threshold. Thus, the parameters (sliding window size, offset size, and quartile threshold) influence had to be analyzed with all the combinations reflected in the visits files. The goal was to keep only the paths with important nodes to the next phase.

Phase 2 ingested the visits files produced at phase 1. Next, we ran the frequent itemsets algorithm and deleted the duplicate entities creating the candidate set. Those sets (from each pair of ontology both networks) were compared using the semantic threshold. The semantic threshold parameter limits candidates by previously comparing some candidates. The final selected candidates from each ontology produced the projection set to phase 3. The setups were statistically compared, and only the best setup was sent to phase 3.

Phase 1 and 2 used as control experiment two baselines: the first select some entities by chance (number more significant than the reference alignment). The second gets entities at random (10x the size of the graph). Both baselines aim to check after phase 2 whether or not the candidates picked up by the baselines have better metrics than our method.

Phase 3 uses the previous results from [41]. There, the ''gray'' (All or Brute Force) and ''blue'' with ''red'' experiment were studied. figure 5. The ''gray'' experiment ran the matchers over all ontologies in both networks running the Cartesian product. The ''blue'' (Naive) experiment ran the union operation [8] creating a single OWL file for each network. The single file was sent to the matchers to retrieve the alignments and metrics. The goal was to evaluate the ability of the state-of-art matchers to deal with a complex set of ontologies, as we have in networks. The ''blue'' experiment can reduce the effort by transforming alignments of the type sigkdd x ekaw and ekaw x sigkdd into only one: sigkdd x ekaw or ekaw x sigkdd. The ''red'' experiment is the SubInterNM implementations running the union, intersection and difference operations defined in [8].

Our study creates two new experiments: the ''green'' and the ''purple.'' The ''green'' (Sub+RW+FIS) experiment uses the results from phase 1 and phase 2 to project the relevant nodes and insert them again in the final network obtained after the SubInterNM approach (extending the ''red'' (Sub) approach). Finally, the ''purple'' (Sub+RW+FIS+Pairwise) approach extends the ''green'' (Sub+RW+FIS) approach by projecting the final union with the fragments of the original ontologies and sending them to the matcher in a pairwise way.

Thus, we verified how the alignments with the fragments compare with the alignments using the original ontologies and how the metrics compare with the Force Brute.

We measured the execution time of each phase of the experiment. For the SubInterNM we got the processing time for each operation. We selected the best setup for the RW and FIS (discovered in phase 2) and ran ten times, calculating the average using each ontology used in the experiment.

The final results were compared with the similar analysis done in [41] as the baseline. All (Brute Force), Naive, and SubInterNM were compared using the LogMap and Alin. Both had good results matching anatomy (Alin and LogMap) and large biomedical ontologies (LogMap) in the OAEI competition [36]. Finally, the parameters influence is discussed in the results section.

D. COMPLEXITY

The complexity analysis was divided into phases. As mentioned in the section III we did not implement the parallelization, but we will mention where it can be used to reduce the time and space complexity as future work.

In phase 1, the random walk procedure runs $r * O(n)$ where r is the number of random walks over the graph structure. In phase 2, the method uses $O((n - s)^2)$ where s is the support threshold used to compare the frequent itemsets created for each pair of entities. However, we can run it without

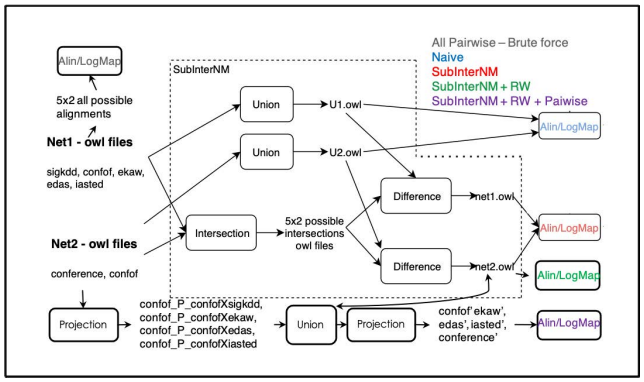

FIGURE 5. Phase 3 - 5 × 2 example.

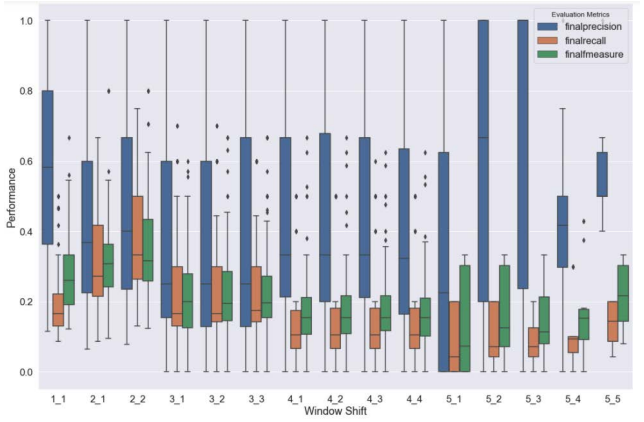

FIGURE 6. Phases 1 and 2 - windows shift comparison.

the semantic threshold over each pair before the alignment. In this case, we can harm the precision by offering more entity entities as candidates since we disregard the target ontology. On the other hand, the frequent itemsets will run with $O(2 * (n - s))$ complexity.

Phase 3, we have many operations: union, intersection, and difference (from the SubInterNM) preceded by the projection (to save the relevant entities discovered by the random walk and frequent itemsets) and finally, the union between the results coming from SubInterNM with the projection.

Casanova et all [8] state the algebraic operation uses $O(n^2)$ in acyclic graphs and is NP-hard when applied to strongly connected graphs. The union uses $O(n^2) * t$ where t is the time to compare the URIs from two different entities. Indeed, the minimization procedures to create the graph and the union can be run before the alignment process. The intersection needs $O(n^2) * t$, and it can be run in parallel, reducing its complexity to $O((n^2/m^2)) * t$ where m is the number of ontologies (supposing both networks have the same number of ontologies).

The difference uses $O((n^2)/2))$ in average since the intersections calculated before are compared with both networks, and each difference can be run in parallel. The projection uses $O(n)$, and the final union complexity is $O((2 * n)/j)$ where j is the number of projected ontologies, and the projection can be run in parallel as well. Finally, the select matchers can align the pruned networks with $O((n - p)^2) * c$ complexity, where p are the pruned entities, and c is the time to calculate all the similarity metrics between a pair of ontologies

The total time complexity equals the most time-consuming step. The most impacting executions are the frequent itemsets comparing entities ($O((n - s)^2)$) in phase 2, $O((n^2)/2))$ from the difference and the time for the matching $O((n - p)^2) * c$ in phase 3.

The Cartesian product to address the pairwise or the holistic approaches uses $O(n^2) * c$.

It is worth mentioning as we have more similar entities in the networks tends to increase the number of entities pruned p and the difference from $O((n - p)^2) * c$ (matching after the method) to $O(n^2) * c$ using the naive Cartesian product. As discussed in the section II-G this is the expected situation when companies merge.

E. PSEUDO CODE

This section summarizes the pseudo code corresponding a each phase of the method 2.

The pseudo-code 1 receives a set of windows sizes and offsets (shifts) and starts producing a random walk from each possible entity e_i^j where $j = \{1, 2\}$ and $i = \{1..n\}$ from possible entities $\in \{O^1, O^2\}$. The random walk checks the adjacent list to select the next node to be visited randomly. For each node visited, a bag of nodes with their paths are stored in B^j. Since bags B^1 and B^2 are created, the quartile check verify whether the path includes all nodes above the quartile limit Q.

The relevant nodes discovery procedure starts using the bags of each random walk created by 1 considering all the window sizes shifts and quartiles output. Using each defined setup (window sizes, shifts and quartiles output), the apriori function returns all the frequent itemsets items F^i, discarding duplicates I^i and calculating the metrics for the results, considering only one ontology regarding the matches with one side of the reference alignment R. From the unique candidates and using the semantic thresholds T defined, we checked the semantic similarity, and those above the threshold are persisted to the next phase E. The proposed matches are returned by the procedure to be used as input in the network matcher optimizer. We also collected the match metrics using both ontologies to analyzed the results M.

The optimized network matcher starts computing the projection for the elements considered relevant E from the algorithm 2 stored in P. The Resulting set of networks Ψ' and each final network Γ' is initialized. Next, for each network and each ontology, we perform the Union operation. This way, all ontologies from a network will be united inside a temporary variable Γ'_i. Following for each network we calculate the Intersection for all ontologies, storing momentarily in $Upsilon_i$. The Difference procedure prunes all intermediate networks Γ'_i with the identified intersections $Upsilon_i$.

Algorithm 1: Random Walk:

Input: Two ontologies $O1$ and $O2$ with a set of entities $E^1 = \{e_1^1, e_2^1, \ldots, e_k^1\}$ and $E^2 = \{e_1^2, e_2^1, \ldots, e_k^2\}$, sliding windows setup window size, shift and quartile (W, O, Q), a set of reference alignments R

Output: Two sets of binary bags B^1, B^2 visited nodes and paths for each window size and shift

```
Random Walk (O^1, O^2, R, W, O, Q)
B^1 ← ∅;
B^2 ← ∅;
while e_i^1 ← 1 to n ∈ O^1 ≠ ∅ ∈ E^1 and e_i^2 ← 1 to
 n ∈ O^2 ≠ ∅ ∈ E^2 do
    while random walk O^1 do
        node ← e_i^1;
        select adjacent node a_i^1 from finite set
         A^1 = {a_1^1, a_2^1, ..., a_n^1} of adjacent matrix of
         node;
        while ∃ path and new window shift do
            V^1 ← visit(e_i^1(a_i^1));
            select adjacent node a_i^1 from finite set
             A^1 = {a_1^1, a_2^1, ..., a_n^1} of adjacent matrix of
             a_i^1;
            i++;
            node ← e_i^1;
        B^1 ← createBag(V^1);
    while random walk O^2 do
        root ← e_i^2;
        select adjacent node a_i^2 from finite set
         A^2 = {a_1^2, a_2^2, ..., a_n^2} of adjacent matrix of
         root;
        while ∃ path and new window shift do
            V^2 ←visit(e_i^2(a_i^2));
            select adjacent node a_i^2 from finite set
             A^2 = {a_1^2, a_2^2, ..., a_n^2} of adjacent matrix of
             a_i^2;
            i++;
            node ← e_i^2;
        B^2 ← createBags(V^2);
for b^1 ∈ B^1 do
    for e^1 ∈ b^1 do
        if e^1 ∉ Q then
            B^1 = B^1 - b^1
for b^2 ∈ B^2 do
    for e^2 ∈ b^2 do
        if e^2 ∉ Q then
            B^2 = B^2 - b^2
Nodes Discovery(B^1, B^2);
```

Finally, the entities saved from the projection are restored using the union operation $\Gamma'_i < -UNION(\Gamma'_i, e_i)$ before calling the matcher passing the final networks Ψ'. A final set of correspondences is returned C.

Algorithm 2: Relevant Nodes Discovery:

Input: Two sets of binary bags B^1, B^2 with visited nodes and paths for each window size, shift, quartile, semantic_threshold T. A reference alignment R

Output: A list of suggested candidate entities $E = \{e_1^1, e_2^1, \ldots, e_k^1\} \in O^1$ x $\{e_1^2, e_2^2, \ldots, e_k^2\} \in O^2$, a set of metrics SM^1, SM^2 and M

```
Nodes Discovery (B^1, B^2)
E ← ∅
for each window size and offset do
    F^1 ← apriori(∀b_i^1 ∈ B^1 )
    for f_i^1 ∈ F^1 ordered by support do
        I^1 ← checkUnicity(f_i^1);
        SM^1 ← calcSideMetrics(I^1);
    F^2 ← apriori(∀b_i^2 ∈ B^2)
    for f_i^2 ∈ F^2 ordered by support do
        I^2 ← checkUnicity(f_i^2);
        SM^2 ← calcSideMetrics(I^2);
    for e_i^1 ∈ I^1 and e_i^1 ∈ R do
        for e_i^2 ∈ I^2 and e_i^2 ∈ R do
            E = E+ ← semantic_check(e^1, e^2, R, T);
            M ← calcFinalMetrics(F, R);
return E, SM^1, SM^2, M;
```

The results section explores the retrieval of the relevant entities, returning to the original pairwise problem. Indeed, explored matchers could not compute alignments when they have more than two ontologies as input. Thus, returning to the pairwise matching can prove the method validity. The pseudo-code above does not contain that step since it was not in our original method and is only an adaptation to show the validity of the proposed methods. Therefore, a set of projections should be executed with the network to retrieve the ontologies fragments, and the projection function is the same presented in the third pseudo code.

V. RESULTS

We present the results grouped by Research question. Phases 1 and 2 will be presented together, and the phase 3 results will be presented separately. Inside phases 1 and 2, we presented the results of the overall approach of the conference domain, mentioning specific characteristics of an experiment involving a pair of ontologies when relevant. The existence of a reference alignment defined those pairs to support the data analysis.

A. RQ1:TO WHAT EXTENT CAN THE METHOD IDENTIFY THE RELEVANT NODES?

The goal of RQ1 is to determine the best setup to feed the network matcher optimizer in phase 3. Therefore we analyzed the results from the random walk and the frequent itemsets.

Algorithm 3: Modified Optimized Network Matching:

Input: A finite set $E = \{e_1, e_2, \ldots, e_n\}$ of entities, two networks of ontologies $\Psi = \{\Gamma_1, \Gamma_2\}$
Output: A list of correspondences $C = \{c_1, c_2, \ldots, c_k\}$

$P \leftarrow \varnothing$;
for $e_i \leftarrow 1$ ***to*** $n \in E$ **do**
  for $\Gamma_i \leftarrow 1$ ***to*** $n \in \Psi$ **do**
    if $e_i \in \Gamma_i$ **then**
      $P \leftarrow P$ + Projection (Γ_i, e_i);
    end if
  end for
end for
$\Psi', \Gamma' \leftarrow \varnothing$;
for $\Gamma_i \leftarrow 1$ ***to*** $n \in \Psi$ **do**
  for $\Omega_i \leftarrow 1$ ***to*** $n \in \Gamma_i$ **do**
    $\Gamma'_i \leftarrow$ Union(Γ'_i, Ω_i);
  end for
  $\Psi' \leftarrow \Psi' + \Gamma'_i$;
end for
Set of intersections $\Upsilon \leftarrow \varnothing$;
for $\Gamma_i \leftarrow 1$ ***to*** $n \in \Psi$ **do**
  for $\Omega_i, \Omega_j \leftarrow 1$ ***to*** $n \in \Gamma_i$ **do**
    $\Upsilon \leftarrow$ Intersection(Ω_i, Ω_j);
  end for
end for
for $\Gamma'_i \leftarrow 1$ ***to*** $n \in \Psi$ **do**
  for $\Upsilon_i \leftarrow 1$ ***to*** $n \in \Upsilon$ **do**
    if $Upsilon_i \in \Gamma'_i$ **then**
      $\Gamma'_i \leftarrow$ Difference$(\Gamma'_i, Upsilon_i)$;
    end if
    $\Psi' \leftarrow (\Gamma'_i)$;
  end for
end for
for $p_i \leftarrow 1$ ***to*** $n \in P$ **do**
  for $\Gamma_i \leftarrow 1$ ***to*** $n \in \Psi$ **do**
    if $p_i \in \Gamma_i$ **then**
      $\Gamma'_i \leftarrow$ Union(Γ'_i, e_i);
    end if
    $\Psi' \leftarrow (\Gamma'_i)$;
  end for
end for
return $C \leftarrow$alignment tool(Ψ');

1) RANDOM WALKS AND FREQUENT ITEMSETS

Our first investigation was about the parameters that could change the dataset results in phases 1 and 2. We started with the configuration of the sliding window parameters: window and shift. The high F-measure results are the window 2 and shift 2 (2-2). The window and shift configuration started losing metrics when the window was beyond the 3 (figure 6). Table 4 shows the statistical comparison between the F-measure from some selected window-shift configurations. Despite the huge variance, it shows the significant difference between the configurations with large Cliff's Delta, except when comparing (2-2) with (1-1) where we found a medium effect size.

TABLE 4. Window and shift configuration comparison (2-2) x (1-1), (3-1), (4-1) and (5-1).

Window	Shift	p-value	Cliff's Delta
1	1	p=0.0000008	0.362, 'medium'
3	1	p<0.00000	0.578, 'large'
4	1	p<0.00000	0.697, 'large'
5	1	p=0.000004	0.655, 'large'

TABLE 5. Threshold precision configuration comparison 0.7 x 0.8 and 0.9.

Metric	Threshold	p-value	Cliff's Delta
Precision	0.8	p<0.7e-19	-0.423, 'medium'
Precision	0.9	p<0.3e-67	-0.753, 'large'
Recall	0.8	p<0.015	0.093, 'negligible'
Recall	0.9	p<0.9e-5	0.161, 'small'

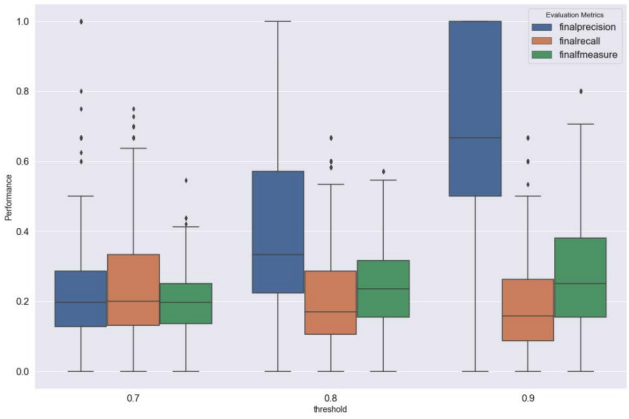

FIGURE 7. Phases 1 and 2 - semantic threshold comparison.

Next, we compared the semantic threshold influence on our results. Looking to the figure 7 we can observe the effect of the semantic threshold on the results. Table 5 shows the Mann-Whitney U test with the Cliff's-delta effect size. Comparing the 0.7 thresholds, 0.8 and 0.9, using precision we found a statistical difference with medium and large effect sizes (0.7 x 0.8 p<0.7e-19, -0.423, 'medium' and 0.7 x 0.9 p<0.3e-67, -0.753, 'large'). Using recall we also found a statistical difference with negligible and small effect sizes (0.7 x 0.8 p<0.015 0.093, 'negligible' and 0.7 x 0.9 p<0.9e-5 0.161, 'small')

We can retrieve better precision by using the 0.9 limits when comparing both candidates from each side (ontologies) and better recall when applying the 0.7 threshold. Figure 9 presents a comparison between setups from iasted and ekaw ontologies. We can see some exceptions to the general rule above. In setup 2-1-1-0.9, the ontologies had opposite behavior. Ekaw (ontology 1) had the best hits while iasted the worst. On the other hand, both had fewer misses. The setup 2-2-2-0.7 has 9 hits (ontology 1 and 2), 32 and 48 misses, respectively.

Finally, we compared the quartile setup from the random walk. The quartile value decides whether a bag from each path created with the sliding window configuration will prevail

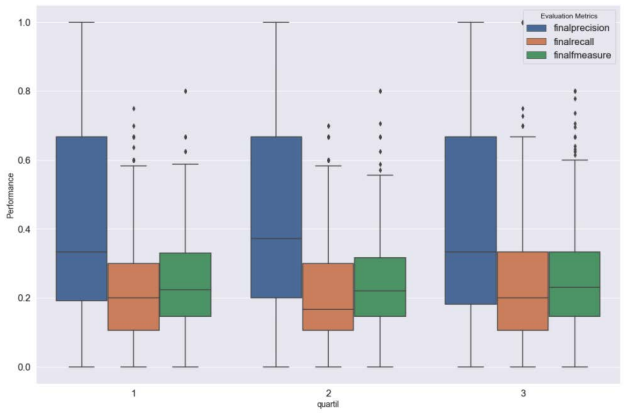

FIGURE 8. Phases 1 and 2 - quartile comparison.

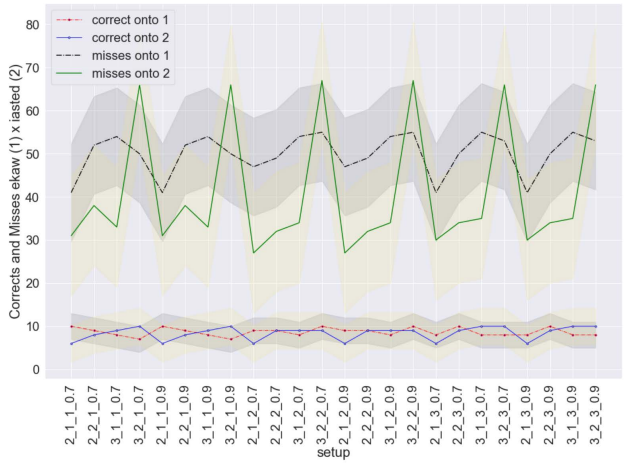

FIGURE 9. Phases 1 and 2 - ekaw - iasted: semantic threshold comparison and standard deviation.

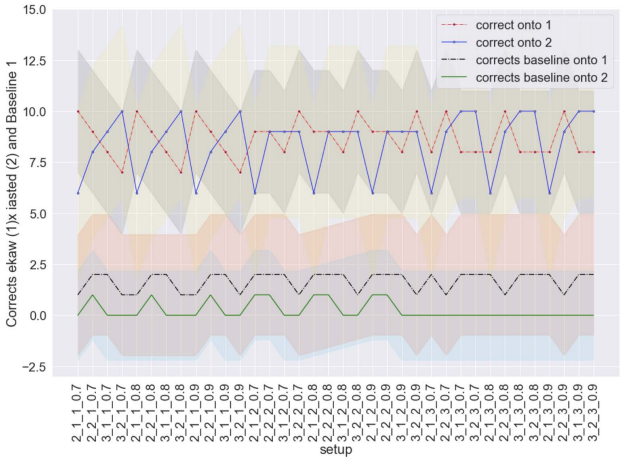

FIGURE 10. Phases 1 and 2 - ekaw - iasted: first baseline corrects comparison and standard deviation for setups.

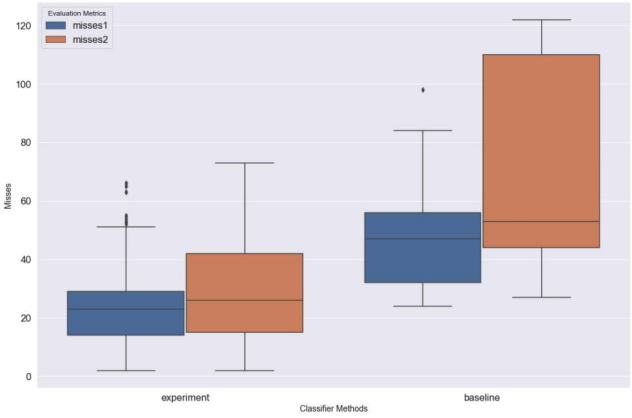

FIGURE 11. Phases 1 and 2 - second baseline misses comparison.

or not. Bags prevail when at least one element has visits beyond the defined quartile, considering the statistics gathered to form all random walks using this ontology. Looking at the figure 8 and running the test, we did not find different distribution, comparing quartiles 1 x 2 and 2 x 3, thus failing to reject H_0 (H_0: quartiles have similar distribution).

We compared phases 1 and 2 with two baselines. Baseline 1 selects a limited number of entities close to the number of entities in the reference alignment to mimic a user guess. The second baseline selected 10x the number of entities randomly to verify the random walk effectiveness to select the relevant nodes based on the graph structure. The results were sent to the apriori algorithm in sequence. Figure 10 shows the comparison of the number of hits with the baseline 1 results. The figure shows one of the bests setups considering the recall in the table 6, since we will send those to the next phase. Selecting some possible alignments close to the number of entities in the reference alignment randomly, as a user behavior probably should do, proved to have worst results than our approach. Next, we aim to verify the random walk performance using the second baseline. Looking at the figure 11 we can see that when selecting the same number of nodes randomly, tends to miss more entities with a statistical difference from the random walk in phase 1 (misses from ontology 1: p<0.8e-20, -0.769, 'large' and misses from the second ontology: p<0.7e-17, -0.701, 'large').

Still measuring the difference from the experiment with the baseline 2, we compared the final precision, final recall, and final F-measure from the experiment and the baseline 2. (p<0.461, effect size: 0.008, 'negligible') (p<0.3e-19, -0.802, 'large') (p<0.9e-10, -0.527, 'large'). This suggests the precision had no statistical difference while having a statistical difference in favor of our approach in recall and F-measure.

Considering the entire conference domain, the ten best recall configuration was the following table 6.

Using the setup (2-2-1-07), we got the hits and misses after the semantic test. The reason to prioritize the best recall is that the final set selected before the semantic test using the defined threshold often suggests the same entities. For instance: ('Organizing_committee', '=', 'Organizing_Committee'), is one select

TABLE 6. Best recall setups.

Experiment	Setup	T	C1	C2	M1	M2	Recall
conference sigkdd	2 2 1	0.7	9	12	38	35	0.750
conference confof	2 2 3	0.7	11	8	10	34	0.727
ekaw iasted	2 2 2	0.7	10	9	9	49	0.700
ekaw iasted	2 2 3	0.7	10	10	9	50	0.700
ekaw iasted	3 1 1	0.7	10	8	9	54	0.700
ekaw iasted	3 1 3	0.7	10	8	10	55	0.700
ekaw iasted	3 2 2	0.7	10	10	9	55	0.700
ekaw iasted	3 2 3	0.7	10	8	10	53	0.700
conference sigkdd	2 1 1	0.7	12	8	12	28	0.667
conference sigkdd	2 1 3	0.7	12	8	12	29	0.667

match considered right bt the reference alignment. However, ('Organizing_committee', '=', 'Committee'), ('Organizing_committee', '=', 'Program_Committee'), ('Organizing_committee', '=', 'Organizing_Committee_member'), are also selected and are wrong matches accordingly with the same reference. The three wrong matches harmed the calculated precision. Therefore, in a first look, they seem a bad choice. Indeed, they are not, since they will be used to project only one entity from their original ontology: 'Organizing_committee.' Thus, many misses in the set of persisted entities may be repeated in pairs of candidate entities. This leads to our recall metrics being underestimated in this moment of the processing pipeline. On the other hand, prioritizing the precision could select a set with only one suggestion and correct, hence with precision = 1.0, thus projecting only one value to use later.

RQ1 Summary. Our findings suggest that it is possible to discover relevant entities using existing statistical data from the ontologies structure. The best setup is the window shift (2-2). The quartile showed no statistical influence, while the semantic threshold increased the recall when it was set up with 0.7. We found that pre-processing the ontology can discover relevant nodes with an average recall of 75%

B. RQ2: TO WHAT EXTENT DO THE RELEVANT ENTITIES IMPROVED THE MATCHING OF NETWORK OF ONTOLOGIES?

Phase 3 carried out the experiments using some network configurations in the table 3 and compared with the results presented in [41] (''blue,'' ''red'' and ''gray'' experiments).

1) GREEN EXPERIMENT

This section presents the results from the ''green'' experiment. Tho goal is to verify whether or not reinserting the projected entities back into the network will improve the alignment metrics obtained in the ''red'' experiment. After running phase 1, random walk, and phase 2, frequent itemsets, we sent the results using the setup 2-2-1-0.7. We calculated the projections from each ontology in the networks that the SubInterNM approach should prune. For instance in the 2 × 2 experiment we matched Ω_1 = {sigkdd, confof} with Ω_2 = {conference, confof}, since 'confof' was pruned, before we had gotten projections from 'confof' ontology using the matching suggestions from phase 2. Hence, 'confof' must be projected with the suggested entities from confof X conference and confof X sigkdd. Therefore, we had confof_P_conferenceXconfof and confof_P_confofXsigkdd operations executed (table 11). The same reasoning applies to the remaining experiments.

TABLE 7. Processing time pairwise - individual cases (seconds.milliseconds).

Experiment	Sub+RW+ FIS+ pairwise +Logmap	Sub+RW+ FIS+ pairwise +Alin	LogMap All	Alin All
conferenceXconfof	**1.791**	6.428	3.997	6.534
conferenceXekaw	**1.880**	6.744	4.419	6.651
confofXsigkdd	**1.749**	5.113	2.067	5.922
confofXedas	2.450	6.095	**2.190**	6.047
confofXekaw	2.315	5.572	**2.108**	6.136
confofXiasted	**2.219**	5.465	2.455	6.401
ekawXsigkdd	**1.907**	5.736	1.991	6.121
ekawXedas	**2.124**	6.452	2.178	6.829
ekawxconfof	**1.777**	5.539	2.108	6.136
ekawXiasted	**2.703**	6.008	3.015	6.386

Next, we run the unions to each network. Since network 1 had 'confof' pruned, we need to make a union operation with the entities suggested as relevant from phase 2 to recover partially, at least, the missing results from 'confof X conference'. Thus, the operation needed is the union between network 1 and the result of the projection using 'confof' and the suggested entities from 'conference X confof' or 'net1_220_U _confof_P_conferenceXconfof'.

To compare the results with the metrics (precision, recall, and F-measure) obtained in [41], we created new rows in the tables starting with 'RW.' The final results were submitted to LogMap [21] and Alin to compute the alignments.

Tables 12 and 8 summarize the metrics and processing time for the ''green'' experiment.

2) PURPLE EXPERIMENT

We can observe in results (from the ''green,'' ''red,'' and the ''blue'' experiment) that the matchers had bad results computing alignments where entities from more than two ontologies are present. Therefore, we run one more projection to split each ontology's entities. The goal of the ''purple'' experiment is to demonstrate the viability of the method and explain why the ''green'' experiment did not overcome the ''red'' one. Indeed, we came back to the pairwise matching, where the matchers had only to ingest a pair of ontologies. The items below show the projection operations executed in the RW2 × 2 experiment. Therefore, we had to compute the execution time used to match the ontology fragments after the projection, as shown in table 7. For instance, from the final result 'net1_220_U _confof_P_conferenceXconfof' and 'net2_220_U _confof_P_confofXsigkdd' we created 'net1220_confof' that is the projection of 'net1_220_U _confof_P_conferenceXconfof' using 'confof'.

- (net1_220_U_confof_P_conferenceXconfof) _P_confof
- (net1_220_U_confof_P_conferenceXconfof) _P_sigkdd
- (net2_220_U_confof_P_confofXsigkdd) _P_confof
- (net2_220_U_confof_P_confofXsigkdd) _P_conference

The results in table 12 confirmed that the matchers are unable to compute alignments using networks of ontologies, probably because of the complexity of the structure. Looking at the lines 2×2 and RW 2×2, the metrics were even worse after the projections. However, when we split the entities in the final projection and send only pairs to the matchers, we overcome the metrics individually and in the average. The 2×2 case is an exception, possibly because the complexity is not relevant to harm the results from the network matching. Also the 2×2 did not computed any correspondences from 'confof' X 'sigkdd' and 'confof' X 'conference'.

The matchers with brute force had better metrics. However, it was expected since the network approaches compare fragments of the ontologies with some missing entities. Some of those entities should possibly be part of the reference alignment. While brute force had better metrics, it needed to compare each entity, while the proposed method avoided many comparisons. After the last projections return the problem to the pairwise matching, the results show the approach is feasible, and the actual matchers must be improved to deal with more than pairs of ontologies at once.

3) PROCESSING TIME ANALYSIS

Next step, we compared the execution time. The goal is to show how the method might be helpful for extensive network matching tasks. Looking at the figure 12 we can compare the execution time-sliced by operation. As expected, we observe that the RW+FIS (RW) and RW+FIS+pairwise (RWPair) times are always higher than the corresponding SubInterNM (Sub) and Brute Force (All). (Experiments operations are showed in figure 5). As expected, the Sub and the All approaches overcame the Sub+RW+FIS and Sub+RW+FIS+Pairwise approaches. The overhead caused by the many preprocessing steps impacted the execution time.

Nevertheless, due to the nature of the experiments and the need to track each intermediate result, manually checking them, our most extensive network had ten nodes, five each network. Figure 12 shows the experiment with force brute (All) uses the matcher (blue bar - LogMap) intensively to compute all the possible alignments. While the interception and union operation can be run in advance, both also can be run in parallel. The difference operations also may start in parallel for both networks, just after the union and interception finish. The RW operation is also independent. RW results may be previously stored based on the best setup for the domain. The FIS also may run without the semantic check and be stored waiting for a future alignment. Even running the FIS with the semantic check, all the semantic checks can be run in parallel. Considering this particular case, we eliminated the union and the RW+FIS to verify how is the behavior of the execution time (figure 13). Considering the experiment 2×2, it is notable the force brute (All) had the second-best time and

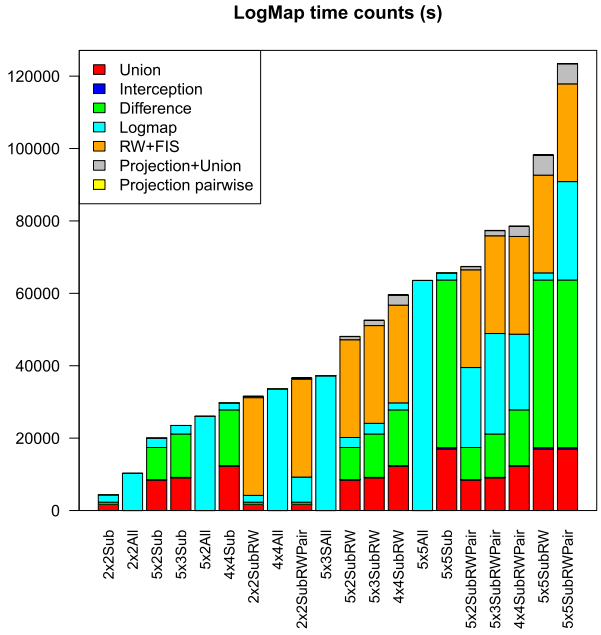

FIGURE 12. Processing time experiments LogMap.

now has the worst time. The experiment 5×5 did not repeat the same results, although the force brute is in the third place overcoming only the RW+FIS+Pairwise (SubRWPair). The projections are pairwise executed in a few milliseconds and are invisible in the figure. Tables 9 and 8 shows the processing time from some experiments. Running the method with previously stored data from the union, RW and FIS helps to decrease the total processing time. Despite saving 27s on average, it's not enough to overcome brute force.

The fastest approach is running the matcher with the entire network grouped with the join ("blue") operation. However, it produces terrible metrics. SubInterNM ("red") is faster than the random walk + frequent itemsets ("green") approach but can miss a lot of pruned entities even if the matcher cannot calculate them as they struggle to deal with more than two ontologies in time. Both struggled to maintain good metrics. RW+FIS+Pairwise ("purple") had metrics close to brute force ("gray"), notably when using Alin 9. On the other hand, the processing time suggests that Alin with RW + FIS outperforms Alin with brute force as networks grow. The same did not happen with LogMap. We limited our experiment to 5×5 networks due to the need to manually check the projections and network joins in the "green" and "purple" experiments. We must predict the results for more extensive networks.

LogMap is clearly faster than Alin and will be used for the predictions.

4) PREDICTION FOR LARGER NETWORK SIZES

Considering the increase in network sizes, we can predict how much those operations will increase execution time. We aim to show that the proposed method may overperform

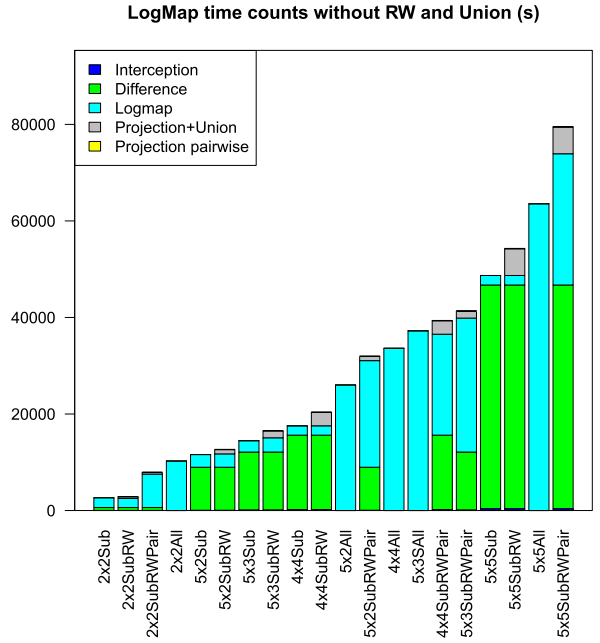

FIGURE 13. Processing time experiments LogMap without Union, Random Walk and Frequent Itemsets.

TABLE 8. Processing Time (seconds.milliseconds).

Experiment	SubInterNM +Logmap	SubInterNM +Alin	LogMap Union	Alin Union
2x2	4.345	8.587	**3.766**	8.710
RW+FIS2x2	4.539	8.221	**3.766**	8.710
4x4	29.376	34.931	**20.541**	22.516
RW+FIS4x4	32.517	37.702	**20.541**	22.516
5x5	65.650	71.494	**19.999**	32.137
RW+FIS5x5	71.179	77.023	**19.999**	32.137
5x1	16.548	21.442	**11.668**	20.537
5x2	20.041	25.151	**11.400**	15.989
RW+FIS5x2	21.058	26.008	**11.400**	15.989
5x3	23.476	29.445	**11.784**	19.445
RW+FIS5x3	25.515	30.595	**11.784**	19.445

TABLE 9. Processing Time Proposed Method (using previously stored RW+FIS data) X Brute Force (All) (seconds.milliseconds).

Experiment	Sub+RW +FIS +Logmap	Sub+RW +FIS +Alin	Logmap All Brute Force	Alin All Brute Force
2x2	**9.628**	19.320	10.274	24.292
4x4	51.560	89.767	**33.615**	88.148
5x5	96.477	146.185	**63.556**	163.343
5x2	40.436	73.004	**26.027**	62.787
5x3	50.386	101.168	**37.238**	94.621

the existing matchers to align the networks by brute force. We ran a regression to observe the evolution of the total execution time in more extensive networks. Our dataset consists of data from the number of nodes (nodes), a total of network comparisons (net1Xnet2All), execution time used by the matcher (LogTime or AlinTime), number of entities

TABLE 10. Predicted processing times (ms). Orange = Purple experiment running some steps in parallel. (Figure 14).

Experiment	SubInterNM	RW + FIS + pairwise (-Union,Diff,RW+FIS)	All (brute force)
6x6	**83,726.28**	**91,392.67**	193,612.7
7x7	**114,235.90**	**127,694.9**	277,705.3
8x8	**148,949.21**	**170,643.9**	372,055.9
9x9	**187,866.22**	**220,239.6**	476,664.7
10x10	**230,986.92**	**276,482.1**	591,531.7
20x20	**893,397.17**	**1,204,476.0**	2,304,400.0
30x30	**1,976,176.93**	**2,797,144.0**	5,043,083.0

comparisons (comparisonsAll), and the total execution time (LogAll or AlinAll).

We run regressions using the faster matcher, LogMap, for the force brute experiment (all), the RW+FIS+Pairwise (without the Union), and the SubInterNM. Our dataset consists of the data gathered in phases 1, 2, and 3. For each experiment, we registered the processing times for the steps. It includes the matcher and more: the number of ontologies to be compared, the number of nodes in networks, and the number of entities being compared. The general regression model used was: lm <- lm(totalExecutionTime $\sim$ networksNodes + ontologiesCompared + entitiesCompared + matcherTime). We found p-values of 0.003, 0.01 and 0.05 respectively and adjusted R-squared of 0.95, 0.77 and 0.74 respectively.

We discarded the entitiesCompared and the matcherTime covariates since we could not reject the null hypothesis $H_0 = entitiesCompared = 0$ and $H_0 = matcherTime = 0$ using the anova test to compare full x reduced model. After we predicted for the selected experiments the total execution time for networks with 6×6, 7×7, 8×8, 9×9 and 10×10, 20×20 and 30×30 nodes, using the predict function. For example: the 6×6 prediction was carried out with the predict(lm, data.frame(ontologiesCompared = 36, networksNodes = 12). Figure 14 shows the graphical comparison with the predicted execution times and the number of nodes. Table 10 shows the predicted processing time for experiments varying from the 6×6 until 30×30 experiment. As the networks grow in size, the proposed approach is faster than the brute force after the number of nodes > 10. The SubInterNM is faster; however, it may suffer from the pruned entities. The method can be even faster identifying the relevant nodes before pruning them, running the union, RW and FIS using previous stored the results and running the difference and the intersections in parallel.

RQ2 Summary. Our findings suggest that finding relevant entities improved the matching with metrics (precision, recall, and F-measure) closer to the brute force approach and faster execution time. The metrics outperformed the SubInterNM method.

VI. DISCUSSION

What is the role of Random Walk in discovering the entity's relevance

TABLE 11. Projection and union operations in some experiments.

Operations/ Experiments	Projections	Unions network 1	Unions network 2
2x2	confof_P_conferenceXconfof confof_P_confofXsigkdd	net1_220 _U_confof_P_conferenceXconfof	net2_220 _U_confof_P_confofXsigkdd
5x5	confof_P_conferenceXconfof, confof_P_confofXsigkdd, confof_P_confofXekaw, confof_P_confofXedas, confof_P_confofXiasted, edas_P_confofXedas, edas_P_conferenceXedas, edas_P_edasXsigkdd, edas_P_edasXekaw, edas_P_edasXiasted, ekaw_P_ekawXiasted, ekaw_P_conferenceXekaw, ekaw_P_confofXekaw, ekaw_P_edasXekaw, ekaw_P_ekawXsigkdd, iasted_P_conferenceXiasted, iasted_P_confofXiasted, iasted_P_edasXiasted, iasted_P_ekawXiasted, iasted_P_sigkddXiasted	net1_550 _U_confof_P_conferenceXconfof _U_confof_P_confofXekaw _U_confof_P_confofXedas _U_confof_P_confofXiasted _U_edas_P_confofXedas _U_edas_P_conferenceXedas _U_edas_P_edasXekaw _U_edas_P_edasXiasted _U_ekaw_P_ekawXiasted _U_ekaw_P_conferenceXekaw _U_ekaw_P_confofXekaw _U_edas_P_edasXekaw _U_iasted_P_conferenceXiaste _U_iasted_P_confofXiasted _U_ iasted_P_edasXiasted _U_iasted_P_ekawXiasted	net2_550 _U_confof_P_confofXsigkdd _U_confof_P_confofXekaw _U_confof_P_confofXedas _U_confof_P_confofXiasted _U_edas_P_confofXedas _U_edas_P_edasXsigkdd _U_edas_P_edasXekaw _U_edas_P_edasXiasted _U_ekaw_P_confofXekaw _U_ekaw_P_edasXekaw _U_ekaw_P_ekawXsigkdd _U_ekaw_P_ekaw_iasted _U_iasted_P_confofXiasted _U_ iasted_P_edasXiasted _U_iasted_P_ekawXiasted _U_iasted_P_sigkddXiasted
5x2	confof_P_conferenceXconfof, confof_P_confofXsigkdd, confof_P_confofXekaw, confof_P_confofXedas, confof_P_confofXiasted	net1_520 _U_confof_P_conferenceXconfof	net2_520 _U_confof_P_confofXsigkdd _U_confof_P_confofXekaw _U_confof_P_confofXedas _U_confof_P_confofXiasted
5x1	xxx	net1_510	net2_510
5x3	confof_P_conferenceXconfof, confof_P_confofXsigkdd, confof_P_confofXekaw, confof_P_confofXedas, confof_P_confofXiasted, ekaw_P_conferenceXekaw, ekaw_P_confofXekaw, ekaw_P_edasXekaw, ekaw_P_ekawXsigkdd, ekaw_P_ekawXiasted	net1_530_ U_confof_P_conferenceXconfof _U_confof_P_confofXekaw _U_ekaw_P_conferenceXekaw _U_ekaw_P_confofXekaw	net2_530 _U_confof_P_confofXsigkdd _U_confof_P_confofXekaw _U_confof_P_confofXedas _U_confof_P_confofXiasted _U_ekaw_P_confofXekaw _U_ekaw_P_edasXekaw _U_ekaw_P_ekawXiasted _U_ekaw_P_ekawXsigkdd

TABLE 12. Precision Recall and F-Measure (RW)+SubInterNM+LogMap (RW)+SubInterNM+Alin LogMap naive Alin naive.

Experiment	SubInterNM +Logmap			SubInterNM +Alin			Logmap - Union			Alin - Union		
	P	R	F	P	R	F	P	R	F	P	R	F
2x2	0,818	0,600	0,692	**0,909**	**0,666**	**0,769**	0,842	0,432	0,571	0,348	0,405	0,375
RW2x2	0,391	**0,600**	0,473	0,024	0,027	0,025	**0,842**	0,432	**0,571**	0,348	0,405	0,375
4x4	0,818	0,600	0,692	**0,909**	**0,666**	**0,769**	0,750	0,141	0,237	0,197	0,192	0,194
RW4x4	0,391	**0,600**	**0,473**	0,024	0,027	0,025	**0,750**	0,141	0,237	0,197	0,192	0,194
5x5	0,818	0,600	0,692	**0,909**	**0,666**	**0,769**	0,583	0,126	0,207	0,096	0,102	0,099
RW5x5	0,391	**0,600**	**0,473**	0,024	0,027	0,025	**0,583**	0,126	0,207	0,096	0,102	0,099
5x1	0,690	0,233	0,348	**0,904**	0,220	0,355	0,778	**0,244**	**0,371**	0,000	0,000	0,000
5x2	0,655	**0,268**	**0,380**	**0,888**	0,225	0,359	0,694	0,164	0,265	0,537	0,132	0,212
RW5x2	0,386	**0,239**	**0,295**	**0,769**	0,136	0,232	0,694	0,164	0,265	0,537	0,132	0,212
5x3	0,833	**0,435**	**0,572**	**0,882**	0,326	0,476	0,674	0,179	0,283	0,300	0,162	0,210
RW5x3	0,217	0,217	0,217	0,224	**0,239**	0,231	**0,674**	0,179	**0,283**	0,300	0,162	0,210

The RW provides a way to understand the ontology's structure through the visited nodes (entities). The connections and centrality of a node tell us the node's relative importance to the structure [23], [45]. The more connected and central a node is, the more probability to be visited. However, there is no guarantee that a relevant node will participate in a reference alignment. Only the data provided by the structure is not enough. The result depends on the semantics between the ontologies to be aligned. The RW seriously impacted the processing time. However, once we have the best setup for a domain of ontologies, We can store the result of the RW and run only the FIS when aware of the ontology to be aligned.

What is the role of the frequent itemsets in discovering the entity's relevance The Frequent itemsets (FIS) compares entities from pairs of ontologies using different RW setups to complement the RW method. The comparisons can be

TABLE 13. Precision recall and f-measure pairwise individual cases.

Experiment	SubInterNM +Logmap			SubInterNM +Alin			Logmap pairwise			Alin pairwise		
	P	R	F	P	R	F	P	R	F	P	R	F
RW+FIS2x2 conferenceXconfof	0,714	0,333	0,454	0,833	0,333	0,476	0,846	0,733	0,785	**0,928**	**0,866**	**0,896**
RW+FIS2x2 confofXsigkdd	**1,000**	0,571	0,727	**1,000**	0,571	0,727	**1,000**	0,714	0,833	**1,000**	**1,000**	**1,000**
RW+FIS5x2 conferenceXconfof	0,714	0,333	0,454	0,833	0,333	0,476	0,846	0,733	0,785	**0,928**	**0,866**	**0,896**
RW+FIS5x2 confofXsigkdd	**1,000**	0,571	0,727	**1,000**	0,714	0,833	**1,000**	0,714	0,833	**1,000**	**1,000**	**1,000**
RW+FIS5x2 confofXedas	0,833	0,263	0,400	0,857	0,315	0,461	0,769	0,526	0,625	**0,928**	**0,684**	**0,787**
RW+FIS5x2 confofXekaw	0,857	0,600	0,706	**1,000**	0,550	0,769	0,933	0,700	0,800	**1,000**	**0,750**	**0,857**
RW+FIS5x2 confofXiasted	0,500	0,333	0,400	**1,000**	0,333	0,500	**1,000**	0,444	0,615	**1,000**	**0,666**	**0,800**
RW+FIS5x3 conferenceXconfof	0,714	0,333	0,454	0,857	0,400	0,545	0,846	0,733	0,785	**0,928**	**0,866**	**0,896**
RW+FIS5x3 conferenceXekaw	0,667	0,400	0,500	0,818	0,360	0,500	0,600	0,480	0,530	**0,888**	**0,640**	**0,744**
RW+FIS5x3 confofXsigkdd	**1,000**	0,571	0,727	**1,000**	0,714	0,833	1,000	0,714	0,833	**1,000**	**1,000**	**1,000**
RW+FIS5x3 confofXedas	0,833	0,263	0,400	0,857	0,315	0,461	0,769	0,526	0,625	**0,928**	**0,684**	**0,787**
RW+FIS5x3 confofXekaw	0,769	0,500	0,606	**1,000**	0,450	0,620	0,933	0,700	0,800	**1,000**	**0,750**	**0,857**
RW+FIS5x3 confofXiasted	0,500	0,333	0,400	**1,000**	0,333	0,500	1,000	0,444	0,615	**1,000**	**0,666**	**0,800**
RW+FIS5x3 ekawXsigkdd	0,875	0,636	0,737	**1,000**	0,636	0,777	0,875	0,636	0,737	**1,000**	**0,727**	**0,842**
RW+FIS5x3 ekawXedas	0,800	0,348	0,485	**0,846**	0,478	0,611	0,750	0,522	0,616	0,823	**0,608**	**0,700**
RW+FIS5x3 ekawxconfof	0,900	0,450	0,600	**1,000**	0,450	0,620	0,933	0,700	0,800	**1,000**	**0,750**	**0,857**
RW+FIS5x3 ekawXiasted	0,875	0,700	0,778	**1,000**	0,700	0,823	0,778	0,700	0,737	**1,000**	**0,700**	**0,823**

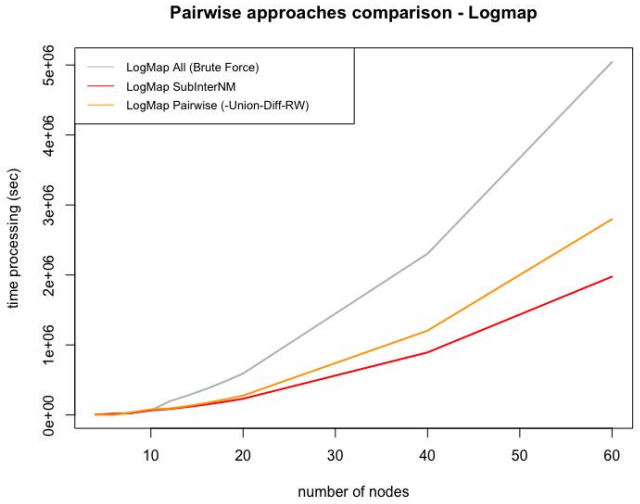

FIGURE 14. Predicted Processing time and number of nodes - Brute Force x SubInterNM x LogMap Pairwise.

made in two ways: with or without the semantic check. The semantic check provides a way to filter out pairs of candidates using a semantic threshold to improve the precision of the persisted pairs. As the RW can select many nodes, this is an important step to avoid using all the discovered ones. The semantic checking, however, poses a performance problem. Since we need to know what ontology we will match, that operation cannot be run in advance to perform the semantic checking. However, it is possible to gather the relevant entities using the RW and FIS without semantic checking. Thus, it permits running in advance. The side effect is a bigger set of candidates improving the recall but harming the precision. Figure 12 show the execution time (average) for each operation. The RW+FIS execution time is in orange. In future work, we can explore to what extent bypassing the semantic check harm the precision.

What metric do we must prioritize

Baseline 2 showed that selecting entities by randomizing the choice can retrieve a representative set of relevant entities. Nevertheless, the price is a poor precision with an extensive set to be used in the projection to increase the execution time in future pipeline steps. The RW and the semantic check provide a way to keep the set smaller than the baseline 2. Interestingly this may indicate that precision should be sought and that this would be the most critical metric. Surprisingly it is not the precision that seems to be the metric to be prioritized when using semantic checking in FIS. When indicating pairs of candidate entities for semantic verification, invariably, an entity often appears in several pairs (ex: (document x conference-document), (document x paper), (document x abstract). Thus, a single indication of the "document" as a candidate entity impacts the precision. However, if we try to prioritize just the precision, dropping all the "document" suggestions, for instance, we will have cases where we will indicate few pairs for phase 3, having an almost perfect precision, close to 1 but with few relevant entities to save from pruning from the subsequent intersection/difference operations. So, tuning the semantic check close to 0.9 had this side effect of discarding too many candidates, getting higher levels of precision, limiting the number of entities in the projection operation too much.

Is the processing overhead worth it

The Random Walk and the Frequent itemsets may run parallel with the SubInterNM, since they depend not on each other. Even the SubInterNM can parallel the Unions and Intersections operations and join together to process the Differences. We ran the projections and the unions (after the SubInterNM) using the ontology manager tab interface and got the average of six operations using the ontologies and partial results. The RW processing time was the average of all domain ontologies. The projections are fast operating, and they will not increase the processing time substantially.

The RW+FIS can be paralleled undependable of the number of ontologies in each network. Thus, the increased number of ontologies will not affect the final processing time.

Also, the proposed method considered three phases using different tools, compared with robust and reliable matchers competing in OAEI many years. The Random Walk (RW), Frequent itemsets (FIS), and the SubInterNM can be integrated into only one application and improve the time processing running all possible steps in parallel.

As the number of nodes increases, the approach reduces the processing time and keeps the metrics closer to the brute force approach. The actual matchers are not prepared to handle alignments with more than two ontologies. Indeed, this limits our proposal, forcing the performance of one more operation to come back to the pairwise comparison. Thus, as soon the matchers can surpass that limitation, the proposed approach will be even better as we can observe comparing the LogMap contribution to the execution time (in light blue) in all experiments SubRW x SubRWPair (figure: 12). Even though the pairwise final operation (in fact, a projection) does not harm the overall execution time, the matcher needs to return to the brute force approach, at least with smaller (pruned) ontologies.

VII. LIMITATIONS

This study is limited to the ontologies from the domain conference. Since the structural approach is used to discover the relevant nodes and the subsequent selection carried out by the frequent itemsets algorithm, which is based on the possible bags, it is supposed to be generalized to other domains once the approach is not tied to the domain characteristics. The variation of the graph shape number of axioms should cause no harm to the RW+FIS. Nevertheless, this suggests an excellent future work to be explored. Another restriction must be taken into the remarks: ontologies like Mouse and Human used by the OAEI initiative should not respond to the semantic check because they have entities with numeric IDs.

The approach is only helpful in cases where we have entities shared in different networks. Networks without entities in common will not benefit from it. Small networks to be aligned or more extensive with small ontologies should not be worth the processing overhead caused by the implemented pipeline. This leads us to another interesting future work: how do we previously know when the approach is worth to be used?

The semantic check implemented can be improved to help better select the candidates. However, improving it can lead us to implement a whole matcher. So, to what extent may the approach be improved before it becomes a matcher?

Even though the predictions showed the method worth it when more extensive networks need to be aligned, many different network shapes and ontologies can vary the results, as probably would see in real cases when one company buys another one and wants to integrate their complex systems. We must investigate it more profound in future work.

VIII. CONCLUSION

In this article, we carry out a case study to investigate the extent to which we can identify relevant entities in the context of ontology networks alignment. To support our research, we needed a set of well-defined ontologies, reliable reference alignments, and proven matchers to carry out our experiments. We use the dataset composed of ontologies from the ''conference'' domain and the matchers participating in the OAEI competitions.

The method prevents relevant concepts to be discarded using the notion of the relative importance of a concept in the ontology's structure. The structural relevancy are discovered by a sampling process using random walks. The results are submitted to an association rule learning algorithm: frequent itemsets.

A previous study pruned some identical entities to save time. We could identify and keep the relevant entities to present them again to the matcher. The method showed significant results compared with the force brute and algebraic approaches, retrieving balanced metrics while presenting a shorter execution time than the force brute approaches for more extensive networks. The SubInterNM was faster but pruned some relevant nodes that may be present in the reference alignment.

In future work, we will test the method with networks of ontologies from diverse domains to verify to what extent the method generalizes. Identifying the relevant nodes before the SubInterNM allow us to not prune the nodes in the difference operation and thus, help to save the processing time.

Three tools were created and available as free, open-source software in the replication package. We employed six ontologies from the conference domain, which are used by the OAEI initiative to compare matcher's performance in our experiments. The results suggest the method's best set up to identify the relevant entities in the conference domain.

The results can be applied to system integration problems when many entities should be compared, avoiding the full Cartesian product (pairwise) check. Incidentally, we discovered that the actual matchers could not retrieve good metrics when comparing more than two ontologies. As the integration problem in the real world should involve more than a couple of systems, the results suggest an opportunity to be explored by researchers working with ontology matching problems.

REFERENCES

[1] *Mlxtend Machine Learning Extensions*. Accessed: May 29, 2021. [Online]. Available: http://rasbt.github.io/mlxtend/
[2] *OAEI Ontology Alignment Evaluation Initiative*. Accessed: May 29, 2021. [Online]. Available: http://oaei.ontologymatching.org/
[3] *Spacy Industrial-Strength NLP*. Accessed: May 29, 2021. [Online]. Available: https://pypi.org/project/spacy/
[4] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, F. Pulvirenti, and L. Venturini, ''Frequent itemsets mining for big data: A comparative analysis,'' *Big Data Res.*, vol. 9, pp. 67–83, Sep. 2017.
[5] B. Boehm, ''A view of 20th and 21st century software engineering,'' in *Proc. 28th Int. Conf. Softw. Eng.*, May 2006, pp. 12–29, doi: 10.1145/1134285.1134288.

[6] M. Bouakkaz, Y. Ouinten, S. Loudcher, and P. Fournier-Viger, "Efficiently mining frequent itemsets applied for textual aggregation," *Int. J. Speech Technol.*, vol. 48, no. 4, pp. 1013–1019, Apr. 2018.

[7] X. Cao, H. Chen, X. Wang, W. Zhang, and Y. Yu, "Neural link prediction over aligned networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 249–256.

[8] M. A. Casanova and R. C. Magalh aes, "Operations over lightweight ontologies and their implementation," in *Implicit and Explicit Semantics Integration in Proof-Based Developments of Discrete Systems*. Tokyo, Japan: Springer, 2020, pp. 61–82.

[9] J. Da Silva, K. Revoredo, F. Baião, and J. Euzenat, "ALIN: Improving interactive ontology matching by interactively revising mapping suggestions," *Knowl. Eng. Rev.*, vol. 35, p. e1 and 22, Jan. 2020.

[10] J. Euzenat, "Revision in networks of ontologies," *Artif. Intell.*, vol. 228, pp. 195–216, Apr. 2015. [Online]. Available: https://ftp.inrialpes.fr/pub/exmo/publications/euzenat2015a.pdf

[11] J. Fitzgerald, S. Foster, C. Ingram, P. G. Larsen, and J. Woodcock, "Model-based engineering for systems of systems: The compass manifesto," COMPASS Interest Group, Tech. Rep. 1, 2013.

[12] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *Int. J. Hum.-Comput. Stud.*, vol. 43, nos. 5–6, pp. 907–928, Nov. 1995.

[13] T. Gruber, L. Ling, and O. M. Tanner, *Ontology Definition Encyclopedia of Database Systems*. New York, NY, USA: Springer Verlag, 2008.

[14] N. Guarino and P. Giaretta, "Ontologies and knowledge bases," *Towards Very Large Knowl. Bases*, vol. 4, pp. 1–2, Oct. 1995.

[15] F. Hamdi, B. Safar, C. Reynaud, and H. Zargayouna, "Alignment-based partitioning of large-scale ontologies," in *Advances in Knowledge Discovery and Management*. Springer, 2010, pp. 251–269.

[16] N. Hareshkumar and D. Garg, "Random web surfer pagerank algorithm," *Int. J. Comput. Appl.*, vol. 975, p. 8887, Oct. 2011.

[17] J. Heaton, "Comparing dataset characteristics that favor the Apriori, Eclat or FP-Growth frequent itemset mining algorithms," in *Proc. SoutheastCon*, 2016, pp. 1–7.

[18] W. Hu, Y. Qu, and G. Cheng, "Matching large ontologies: A divide-and-conquer approach," *Data Knowl. Eng.*, vol. 67, no. 1, pp. 140–160, 2008.

[19] H. Jeong and B.-J. Yoon, "Accurate multiple network alignment through context-sensitive random walk," *BMC Syst. Biol.*, vol. 9, no. 1, p. S7, 2015, doi: 10.1186/1752-0509-9-S1-S7.

[20] E. Jiménez-Ruiz, "Logmap family participation in the OAEI 2019," in *Proc. CEUR Workshop*, 2019, pp. 1–4.

[21] E. Jiménez-Ruiz and G. B. Cuenca, "LogMap: Logic-based and scalable ontology matching," in *Proc. Int. Semantic Web Conf.* Bonn, Germany: Springer, 2011, pp. 273–288.

[22] K. Kalecky and Y.-R. Cho, "PrimAlign: PageRank-inspired Markovian alignment for large biological networks," *Bioinformatics*, vol. 34, no. 13, pp. i537–i546, Jul. 2018, doi: 10.1093/bioinformatics/bty288.

[23] K. Kempf-Leonard, "Encyclopedia of social measurement," Tech. Rep., 2004.

[24] W. Kuánierczyk, "Taxonomy-based partitioning of the gene ontology," *J. Biomed. Informat.*, vol. 41, no. 2, pp. 282–292, Apr. 2008.

[25] S. Lambrini and K. Achilles, "Composable relations induced in networks of aligned ontologies: A category theoretic approach," *Axiomathes*, vol. 25, no. 3, pp. 285–311, Sep. 2015.

[26] M. Lenzerini, "Data integration: A theoretical perspective," in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2002, pp. 233–246.

[27] L. Lovász, "Random walks on graphs," *Combinatorics*, vol. 2, pp. 1–46, Apr. 1993.

[28] R. C. Magalhães, M. A. Casanova, B. P. Nunes, and G. R. Lopes, "On the implementation of an algebra of lightweight ontologies," in *Proc. 21st Int. Database Eng. Appl. Symp.*, 2017, pp. 169–175.

[29] M. W. Maier, "Architecting principles for systems-of-systems," *Syst. Eng.*, vol. 1, no. 4, pp. 267–284, 1998.

[30] N. Masuda, M. A. Porter, and R. Lambiotte, "Random walks and diffusion on networks," *Phys. Rep.*, vols. 716–717, pp. 1–58, Nov. 2017.

[31] I. O. B. Mountasser and B. Frikh, "Parallel Markov-based clustering strategy for large-scale ontology partitioning," in *Proc. KEOD*, 2017, pp. 195–202.

[32] R. Muliono, Muhathir, N. Khairina, and M. K. Harahap, "Analysis of frequent itemsets mining algorithm againts models of different datasets," *J. Phys., Conf. Ser.*, vol. 1361, no. 1, Nov. 2019, Art. no. 012036.

[33] N. F. Noy and M. A. Musen, "Specifying ontology views by traversal," in *Proc. 3rd Int. Semantic Web Conf.*, vol. 3298. Hiroshima, Japan: Springer, 2004, pp. 713–725.

[34] P. Ochieng and S. Kyanda, "Large-scale ontology matching: State-of-the-art analysis," *ACM Comput. Surveys*, vol. 51, no. 4, pp. 1–35, Jul. 2019.

[35] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep. 422, 1999.

[36] M. A. N. Pour, "Results of the ontology alignment evaluation initiative 2021," in *Proc. 16th Int. Workshop Ontol. Matching Co-Located*, vol. 3063, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, Eds., 2021, pp. 62–108. [Online]. Available: http://ceur-ws.org/Vol-3063/oaei21_paper0.pdf

[37] E. Rahm, "Towards large-scale schema and ontology matching," in *Schema Matching Mapping*. Berlin, Germany: Springer, 2011, pp. 3–27.

[38] J. Romano, J. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using T-test and Cohen'sd for evaluating group differences on the NSSE and other surveys," in *Proc. Annu. Meeting Florida Assoc. Inst. Res.*, 2006, pp. 1–3.

[39] F. Santos, K. Revoredo, and A. F. Bai, "Paving a research roadmap on network of ontologies," in *Proc. 12th Int. Workshop Ontol. Matching Co-Located 16th Int. Semantic Web Conf. (ISWC)*, 2017, vol. 7, no. 4, pp. 396–420.

[40] F. Santos, K. Revoredo, and F. Baiao, "A proposal for optimizing internetwork matching of ontologies," in *Proc. ISWC Workshop*, 2018, p. 71.

[41] F. Santos, K. Revoredo, and F. Baiao, "SUBINTERNM: Optimizing the matching of networks of ontologies," in *Proc. Matching*, 2020, p. 77.

[42] A. Schlicht and H. Stuckenschmidt, "Criteria-based partitioning of large ontologies," in *Proc. 4th Int. Conf. Knowl. Capture*, 2007, pp. 171–172.

[43] M. H. Seddiqui and M. Aono, "An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size," *J. Web Semantics*, vol. 7, no. 4, pp. 344–356, Dec. 2009.

[44] P. Shvaiko and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 158–176, Jan. 2013.

[45] Y. Wang, Z. Di, and Y. Fan, "Identifying and characterizing nodes important to community structure using the spectrum of the graph," *PLoS ONE*, vol. 6, no. 11, Nov. 2011, Art. no. e27418.

[46] C. Xiang, B. Chang, and Z. Sui, "An ontology matching approach based on affinity-preserving random walks," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1–7.

[47] M. Seddiqui, *Case Study Research: Design and Methods Applied Social Research Methods Series*. Beverly Hills, CA, USA: Sage, 1984.

FABIO SANTOS received the B.S. and M.S. degrees in informatics (databases) from the Pontifícia Universidade Católica do Rio de Janeiro, Brazil, in 2002. He is currently pursuing the Ph.D. degree in computer science (information systems, knowledge modeling and reasoning) with the Universidade Federal do Estado do Rio de Janeiro, Brazil. He was developer, DBA, project manager, and IT Superintendent in Brazilian Navy. His research interest includes the study of knowledge modeling to support integration of system of systems and network of ontologies.

CARLOS E. MELLO received the M.Sc. degree in engineering from the Universidade Federal do Estado do Rio de Janeiro, Brazil, in 2008, and the Ph.D. degree in computer science from the École Centrale Paris, France, in 2013. He is currently an Associate Professor at the Universidade Federal do Estado do Rio de Janeiro with more than ten years conducting data science projects. He has advised and co-advised more than 15 graduate students, among master's and Ph.D. In recent years, he has been focused on developing data science for social good projects with the Brazilian government and private sector. He leads the Research Group on Data Science for Social Welfare at the University of Rio de Janeiro (UNIRIO).