



Ontology Modularization with OAPT

Alsayed Algergawy¹ · Samira Babalou¹ · Friederike Klan² · Birgitta König-Ries¹

Received: 1 June 2018 / Revised: 23 December 2019 / Accepted: 8 August 2020 / Published online: 24 August 2020
© The Author(s) 2020

Abstract

Ontologies are the backbone of the Semantic Web. As a result, the number of existing ontologies and the number of topics covered by them has increased considerably. With this, reusing these ontologies becomes preferable to constructing new ontologies from scratch. However, a user might be interested in a part and/or a set of parts of a given ontology, only. Therefore, ontology modularization, i.e., splitting up an ontology into smaller parts that can be independently used, becomes a necessity. In this paper, we introduce a new approach to partition ontology based on the *seeding-based* scheme, which is developed and implemented through the Ontology Analysis and Partitioning Tool (*OAPT*). This tool proceeds according to the following methodology: first, before a candidate ontology is partitioned, *OAPT* optionally analyzes the input ontology to determine, if this ontology is worth considering using a predefined set of criteria that quantify the semantic and structural richness of the ontology. After that, we apply the seeding-based partitioning algorithm to modularize it into a set of modules. To decide upon a suitable number of modules that will be generated by partitioning the ontology, we provide the user a recommendation based on an information theoretic model selection method. We demonstrate the effectiveness of the *OAPT* tool and validate the performance of the partitioning approach by conducting an extensive set of experiments. The results prove the quality and the efficiency of the proposed tool.

1 Introduction

Ontologies are the backbone of the Semantic Web, which provides facilities for integrating, searching, and sharing information on the Web by making it understandable for machines [25,26]. According to a study by *d'Aquin et al.* [10] already in 2007, at least 7000 ontologies existed in the Semantic Web, providing an unprecedented set of resources

for developers of semantic applications. However, this large number of available ontologies makes it hard for users to determine which ontologies suitable for their needs. Even, if the user settles on an ontology (or a set of ontologies), she might be interested in a subset of concepts of the ontology, only. For example, if a user plans to use the CHEBI ontology,¹ with 132,082² fully annotated concepts, she is typically not interested in all concepts of the ontology, but just a few of them.

Electronic supplementary material The online version of this article (<https://doi.org/10.1007/s13740-020-00114-7>) contains supplementary material, which is available to authorized users.

✉ Alsayed Algergawy
alsayed.algergawy@uni-jena.de

Samira Babalou
samira.babalou@uni-jena.de

Friederike Klan
Friederike.Klan@dlr.de

Birgitta König-Ries
birgitta.konig-Ries@uni-jena.de

In this situation, in order to create a knowledge base for a specific scenario, the developer has two options: creating proper ontologies from scratch or reusing existing ones. The first option is a time-consuming and labor-intensive process. The maintenance, reuse, and integration of existing ontologies, on the other hand, are also highly complex tasks. In particular, extracting only relevant parts from ontologies that often contain thousands of concepts is a key challenge. Ontology modularization can be used to support this task [11,31,43,49]. In general, modularization approaches can be classified into two main categories: *module extraction* and *module partitioning*. An ontology module is defined as a reusable

¹ Heinz-Nixdorf Chair for Distributed Information Systems, Institute for Computer Science, Friedrich Schiller University of Jena, Jena, Germany

² Institute of Data Science, German Aerospace Center, Jena, Germany

¹ <https://www.ebi.ac.uk/chebi/>.

² information retrieved from <https://biportal.bioontology.org/ontologies/CHEBI> on 07.05.2019.

component of a larger or more complex ontology [12,43], which is self-contained but bears a definite association to other ontology modules, including the original ontology. Ontology partitioning aims at grouping entities with similar characteristics together in order to facilitate their further processing. It has been used for different applications: ontology matching [1,4,27], knowledge selection [11], and reasoning [24,45]. Module extraction extracts from the given ontology a small fragment that captures the intended meaning of the input terms [7,17], while ontology partitioning splits the given ontology into a set of modules [1,20,51]. In this paper, we focus on the second problem: *ontology partitioning*.

The main challenge of ontology modularization is to balance between modularization quality and modularization efficiency. To cope with these challenges, we introduce the Ontology Analysis and Partitioning Tool (*OAPT*), a framework for analyzing and partitioning ontologies. Before partitioning an ontology, the tool optionally investigates the semantic and structural richness of input ontologies using a predefined set of quality criteria. Once an ontology has been selected, the partitioning algorithm is applied to partition it into a set of modules. The partition algorithm is based on a seeding-based clustering approach, called *SeeCOnt*, which starts by determining and identifying which ontology concepts can potentially be used as cluster seeds, called *cluster heads*, *CHs*. To achieve this goal, we develop a new ranking function that quantifies the importance of ontology concepts based on their contexts. In the next step, *OAPT* determines how many of these important concepts should be actually used as cluster heads, i.e., the most suitable number of modules. This step adopts an information theoretic selection method. The proposed method considerably reduces the required effort compared to the naive approach which simply tests all possible number of modules from one to the number of concepts in the ontology. After settling on the desirable number of modules, we apply the *SeeCOnt* algorithm to assign the remaining concepts to the proper cluster head, creating a set of disjoint partitions. Finally, we obtain the set of output modules, by constructing a module for each partition. In contrast to partitions, modules are not necessarily disjoint: we allow for some concepts to be shared across different modules in order to preserve the knowledge residing in the original ontology. To validate the proposed approach, we carried out an extensive set of experiments. We collected a set of the ontologies from BioPortal.³ These ontologies have different characteristics and represent different domains. The experimental results show that the proposed approach performs well w.r.t. a set of criteria that validate the trade-off between the modularization quality and the modularization efficiency. A qualitative user evaluation confirms the usefulness

of the proposed approach but also shows some areas for improvement.

To sum up, the main contributions of this work are:

- Introducing a set of quality criteria to investigate and analyze the semantic and structural richness of ontologies before reuse,
- Introducing a new seeding-based approach to partition the selected ontology,
- Employing an information-theory-based method to recommend the desirable number of modules to partition the ontology
- Developing and implementing a tool to analyze and modularize ontologies, called *OAPT*,
- Conducting a set of experiments to validate different components of the tool.

The remainder of the paper is organized as follows: background and related work are presented in Sect. 2. Section 3 provides an overview of the proposed tool. Detailed descriptions of different components of the tool are introduced in Sect. 4. The performance of the proposed *OAPT* tool is investigated in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Background

This section is devoted to first present basic preliminaries that are used throughout the paper, and then to introduce related work.

2.1 Definitions

Definition 1 An ontology, \mathcal{O} , is defined as a formal, explicit specification of a shared conceptualization [22,59]. We describe an ontology as a 6-tuple, denoted as $\mathcal{O} = \{C, P, H^C, H^P, A, I\}$. C and P are two disjoint sets of classes (concepts) and properties, respectively. $H^C = \{(C_1, C_2) \in C \times C \mid C_1, C_2 \in C\}$ represents the hierarchy of class subsumption. Similarly, H^P is the hierarchy between properties. A is a set of axioms and I is a set of instances associated with the set concepts C and properties P .

The set of properties includes two subsets: P_b is the built-in properties, such as `rdfs:domain` and `rdfs:range` and P_u is the user-defined properties, i.e., $P = P_b \cup P_u$. A signature \mathcal{S} of an ontology \mathcal{O} based on a description logic \mathcal{L} is the union of concepts, properties, and instances, i.e., $\mathcal{S} = C \cup P \cup I$.

In general, ontology modularization covers the problem of identifying a fragment or a set of fragments of an ontology. The process of identifying a fragment of an ontology given a user input (request) is called *ontology module extraction* [17,

³ <http://bioportal.bioontology.org/>.

48,49], while the process that partitions the ontology into a set of fragments is called *ontology partitioning* [2,6,45]. In this work, we distinguish between an *ontology partition* and *ontology module*. We see the ontology partition as a subset of the ontology concepts, (i.e., $\subseteq C$), while the ontology module can be defined as a reusable part of the ontology, which is self-contained but bears a definite association with other ontology modules, including the original ontology [12]. Formally, we define an *ontology module* $\mathcal{M}_i(\mathcal{O})$ following [17,19,32]:

Definition 2 A module $\mathcal{M}_i(\mathcal{O})$ is a module of the ontology \mathcal{O} w.r.t. a description logic \mathcal{L} , if for every axiom α over \mathcal{L} with $\mathcal{S}(\alpha) \subseteq \mathcal{S}$, we have $\mathcal{M}_i(\mathcal{O}) \models \alpha$ i f $\mathcal{O} \models \alpha$.

An ontology module can be represented as a 6-tuple $\mathcal{M}_i(\mathcal{O}) = \{C_{\mathcal{M}_i}, P_{\mathcal{M}_i}, H_{\mathcal{M}_i}^C, H_{\mathcal{M}_i}^P, A_{\mathcal{M}_i}, I_{\mathcal{M}_i}\}$, where $C_{\mathcal{M}_i} \subseteq C$, $P_{\mathcal{M}_i} \subseteq P$, etc. This definition implies that any information that exists or can be entailed from the module $\mathcal{M}_i(\mathcal{O})$ should also exist or could be entailed from the original ontology \mathcal{O} . This enables reusing of ontology modules either as they are or by enlarging them by adding more concepts and relations. Therefore, each module can be considered as an ontology by itself. To achieve this, each ontology module should be *self-contained*, *consistent*, and *topic-centric* [6,12,58].

To support dealing with different ontology formats, we represent each ontology internally as a concept graph. We define a *concept graph* as follows:

Definition 3 A *concept graph* $\mathcal{G} = (\mathcal{N}, \mathcal{R}, \mathcal{LAB})$ is defined as a labeled directed graph, where

- $\mathcal{N} = \{n_1, n_2, \dots, n_n\}$ is a finite set of nodes presenting the entities of the ontology, i.e., concepts, object properties, and instances.
- $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$ stands for a finite set of directed edges showing various relationships between entities in an ontology \mathcal{O} , such that $r_k \in \mathcal{R}$ represents a directed relation between two adjacent entities, and
- \mathcal{LAB} is a finite set of labels of graph nodes and graph edges defining the features of each entity, such as the names of concepts and the name of a relationship between two entities.

To construct such a concept graph from the ontology, we make use of a set of the rules similar to those rules in [27].

We define the ontology modularization process (partitioning) as follows: given an ontology \mathcal{O} represented as a concept graph \mathcal{G} , partition concepts \mathcal{C} of the graph into a set of modules $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$ such that the cohesion of concepts in each module is high (i.e., intra-module distance is low), while the coupling between any pair of modules is low. To achieve this goal, we make use of the following set of criteria, which

have been designed as a trade-off between the modularization quality and the modularization efficiency [3,50]. The first four criteria were originally introduced in [50], while the last one was developed in our own previous work [3].

- **Size.** Given an ontology \mathcal{O} which has been partitioned into \mathcal{K} modules $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$, the relative size of any module pair $\mathcal{M}_i, \mathcal{M}_j$ should be within an acceptable range, i.e., $\frac{|\mathcal{M}_i|}{|\mathcal{M}_j|} \geq \epsilon$, where \mathcal{M}_i and \mathcal{M}_j are the two modules with the minimum and the maximum number of concepts, respectively. The size of a module has a strong influence on its maintainability and on the robustness of the applications relying on it.
- **Correctness.** A module \mathcal{M}_i of an ontology \mathcal{O} should contain only the information that is present in \mathcal{O} . This means that any knowledge that can be inferred from \mathcal{M}_i , should also be inferrable from \mathcal{O} .
- **Completeness.** The generated set of modules should preserve the original ontology, i.e., $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \dots \cup \mathcal{M}_k = \mathcal{O}$
- **Connectedness.** The original ontology is represented as a concept graph, a module \mathcal{M}_i of an ontology \mathcal{O} can also be represented as an independent concept graph. The connectedness of a module is then evaluated on the basis of the number of edges it shares with other modules. That means that a good module should have low inter-related connections to other modules.
- **Distance.** Given an ontology \mathcal{O} modularized into a set of modules $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$, it is worth to measure intra-module distance as well as inter-module distance. The intra-module distance (module cohesion) is to quantify how the concepts in a module move closer to each other as an indication of the homogeneous structure of the module. The inter-module distance (module coupling) is to determine the number of modules that have to be considered to relate two entities.

2.2 Related Work

The proposed tool will address several aspects with respect to ontologies. It first starts by analyzing candidate input ontologies in order to check their semantic and structural richness for ontology reuse, then the chosen ontology will be modularized into a set of modules, where the number of modules will be recommended by an information-theory-based approach. Along this line of thinking, the related work section covers these aspects: *ontology analysis*, *ontology modularization/partitioning*, and *selection of an optimal number of modules*.

2.2.1 Ontology Analysis

Ontology analysis is a well recognized and useful technique to facilitate ontology understanding and then support ontol-

ogy reuse. There are a number of efforts that apply different criteria, addressing different features of ontologies, to extract ontology summaries [34–36,46,47,54,60]. The *OntoEnrich* web platform is used to analyze and detect ontology lexical regularities which might help the detection of hidden semantics. The platform provides a series of useful observations about the structure and content of the concepts' labels, which can be helpful for the study of the engineering of the ontology and their axiomatic enrichment [46,47]. The framework proposed in [36] first loads and parses the input ontology and transforms it into a semantic derived model (SDM). Furthermore, the framework analyzes and summarizes the measurement entities, and an approach is developed to collect and calculate these entities based on the semantic derived model.

OntoMetrics is an ontology evaluation tool used to automatically calculate a set of indicators that can be used to assess ontology quality. Readability, adaptability, and reusability are examples of these indicators [34]. The tool defines five dimensions for description of metric-based ontology evaluation: ontology scopes, ontology layers, ontology life-cycle, ontology quality criteria, and ontology. Two examples are provided to illustrate the usage scenarios: ontology design pattern and analysis of enterprise architecture languages and models [34]. *OntoQA* is another tool that is used to analyze ontologies based on a set of metrics [60]. These metrics emphasize key features of an ontology schema as well as its population. The set of metrics covers both schema and instance dimensions. Within each dimension, specific metrics have been proposed to evaluate the quality of an ontology with respect to this dimension. Therefore, *OntoQA* can be considered as a useful tool for ontology users before considering an ontology as a source of information. A recent study is introduced to describe different use cases when selecting a unit ontology to a specific application [56]. The study presents a different groups for these cases, such as data annotation, conversion, and consistency checking.

While there are many approaches for ontology analysis, each of them has been developed for a specific context and a specific case study. To the best of our knowledge, no ontology analysis tool for the ontology modularization context exists. Therefore, we introduce an ontology analysis method that can be used specifically in the context of ontology modularization.

2.2.2 Ontology Modularization

Ontology modularization is a valuable solution to overcome challenges of large ontologies such as maintainability, reusability, complexity, and scalability. The modularization process is motivated by the problem of dealing with complex and large-scale ontologies by decomposing them into modules. Therefore, many ontology modularization approaches

have been proposed, and several prototypes have been developed [7,8,17,43,61,63]. These approaches can be classified into two main categories: *ontology module extraction* and *ontology partitioning*. In each category, either *structure-based* or *logic-based* approaches can be used. The logic-based approaches develop formal algorithms based on sound logical foundations for module extraction that are correct and complete [17–19,48]. The structure-based approaches present simpler algorithms to modularize an ontology by traversing the ontology hierarchy, and applying heuristic rules to identify the set of sub-graphs [7,12,53].

Several approaches aim at ontology modularization with the goal of partitioning ontologies into a set of modules. *SWOOP* is a web-based OWL ontology browser and editor [29]. The tool offers a set of different ontology operations, such as OWL validation, various OWL representation syntax views, and ontology partitioning. It partitions an ontology into a set of modules connected by ϵ – *connections* [21]. It aims at preserving the completeness of local reasoning within all created modules. This requirement is supposed to make the approach suitable for supporting selective use and reuse since every module can be exploited independently of the others. *ModOnt* is a modularization tool proposed based on a graph/hierarchy traversal approach [12]. The tool inherits some of the main principles from object-oriented software engineering, such as encapsulation and information hiding. The *PATO* tool is a tool used to partition large ontologies into smaller modules based on the structure of the class hierarchy [51]. The approach has three main steps: first, it creates a dependency graph corresponding to the subclass hierarchy; it then measures the strength of these dependencies; and finally, it determines modules. Aquin et al. [11] propose an ontology modularization framework based also on graph transformation. A recent survey aims to identify and locate dimensions of modules, experimentally evaluate and characterize 189 existing modules, and create a framework for modularity based on these results [31]. The work in [62] introduces large-scale investigation into decomposability and modular aspects of the NCBO BioPortal ontologies and demonstrates that most of them can be split into small logically coherent parts, called *atoms*.

Extracting a module by a logic-based approach was proposed by Grau et al. The authors propose a definition of a module that guarantees to completely capture the meaning of a given set of terms, i.e., to include all axioms relevant to the meaning of these terms, and study the problem of extracting minimal modules [17,19]. They use two approximations: the first approximation is semantic and can be computed using existing DL reasoners; the second is syntactic.

Our modularization approach is a partitioning-based approach, however, it differs from the current approaches in the following: we analyze input ontologies before modularizing in order to extract some useful facts that help understand

the behavior of the partitioning process. Furthermore, it is a seeding-based approach, which supports modularization of very large ontologies, and we recommend the optimal number of modules that an ontology should be partitioned into.

2.2.3 K-selection

It is obvious that the number of modules, \mathcal{K} , plays a crucial role in ontology partitioning and affects the performance of ontology modularization. In general, the number of modules (partitions) is an unknown parameter which needs to be either specified by users based on their prior knowledge or estimated by the program. Since partitioning shows similarities to clustering, it is worthwhile to have look at how clustering approaches determine the number of clusters. A variety of methods have been proposed to estimate the number of clusters [33,38,44,65]. In the following, we present a summary of these methods.

- Values of \mathcal{K} specified by the user. Many data mining and data analysis softwares require the number of partitions to be specified by the user. To find a satisfactory clustering result, usually, a number of iterations are needed where the user executes the algorithm with different values of \mathcal{K} .
- Values of \mathcal{K} specified within a range. The performance of a clustering algorithm may be affected by the chosen value of \mathcal{K} . Therefore, instead of using a single predefined \mathcal{K} , a set of values might be adopted.
- Values of \mathcal{K} determined by statistical methods. There are several statistical measures available for selecting \mathcal{K} . These measures are often applied in combination with probabilistic clustering approaches. They are calculated with certain assumptions about the underlying distribution of the data. The Bayesian information criterion (BIC) or Akaike information criterion (AIC) is calculated on data sets which are constructed by a set of Gaussian distributions [28].
- Values of \mathcal{K} determined through visualization. Visual verification is applied widely because of its simplicity and explanation possibilities. Visual examples are often used to illustrate the drawbacks of an algorithm or to present the expected clustering results [65].
- Values of \mathcal{K} determined in a later process. When K-means clustering is used as a preprocessing tool, the number of clusters is determined by the specific requirements of the main processing algorithm [65].

In the proposed framework, we provide two different possibilities depending on the user experience. First, if the user has sufficient experience with the input ontology, she can specify the number of modules, otherwise, she can ask the tool to recommend a suitable number of modules.

3 The Proposed Approach: An Overview

The main objective of the proposed framework is to partition a given ontology into a set of modules after investigating whether this ontology is worthy to be reused. To achieve this objective, we developed and implemented an ontology analysis and partitioning tool, (*OAPT*), consisting of a set of components, as shown in Fig. 1. In this section, we give an overview about these components and their functionalities, while we provide more technical details about them in the next section.

3.1 Preprocessing Component

The main goal of this component is to allow the proposed approach be able to cope with ontologies represented in different formats and languages. To this end, each input ontology is checked to identify its format, and a format conversion is applied when necessary. After that, input ontologies are parsed and then represented internally as concepts graphs.

3.2 Analysis Component

After an input ontology is parsed and represented as a concept graph, the *OAPT* tool analyzes the candidate ontology based on a predefined set of criteria. There are two main objectives behind using this analysis component: the first is to direct our proposed partitioning algorithm to be adaptive based on the internal characteristics of the input ontology, and the second is to extract some facts from ontology analysis to be able to understand the behavior of ontology partitioning. To this end, we first promote collecting relevant information that can be used to validate and evaluate the quality of ontologies. As it is known that the way an ontology is engineered is largely based on the domain in which it is designed and modeled, the ontology design and its potential to represent knowledge should be examined [60]. To cover these different aspects, we classify required relevant information for analysis into three categories: *structural*, *semantic*, and *syntactic*. For each category, we propose a richness metric to calculate the ontology richness with respect to this dimension. Finally, we use the total ontology richness as a metric for its quality. To this end, we make use of *ontology design metric*, as an indicator for the structural dimension, *knowledge-base metric* and *class metric* for the semantic and syntactic metrics as follows:

- **Design metric** This dimension describes the topology of the concept hierarchy of an ontology. It includes several criteria, such as relationship, attribute, depth richness. The **relation richness, RR**, reflects the variability in types of relations and placement of relations in the ontology. An ontology that contains numerous relation types other than class–subclass (is-a) relations is richer than

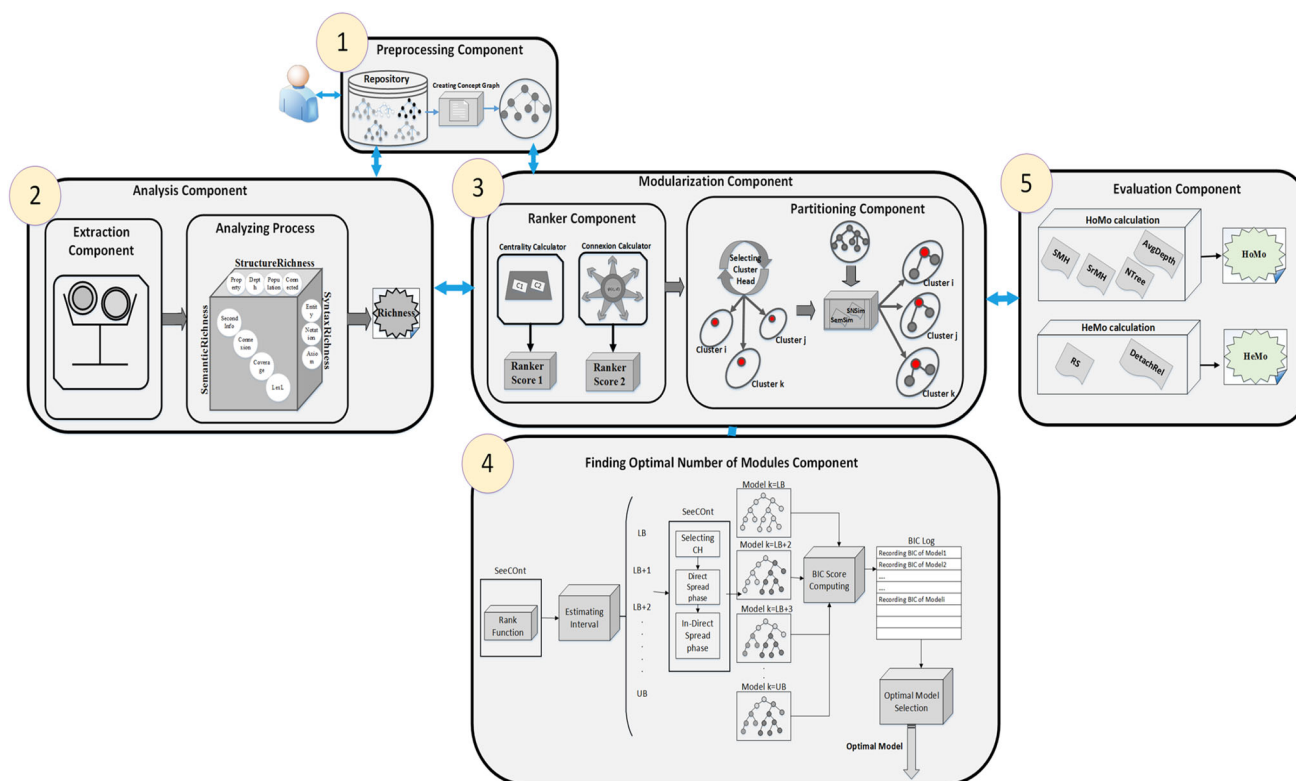


Fig. 1 Proposed tool architecture

a taxonomy with just class–subclass relations. The relation richness (RR) can be defined as: $RR(\mathcal{O}) = \frac{|R \setminus H^C|}{|R|}$, where $R = H^C \cup H^P$ is the set of all relationships in the ontology, and $|H^C|$ is the number of subclass (is-a) relations. The value of the relation richness criterion is normalized between 0 and 1, where the value of 0 means that the ontology contains only subclass relationships. Another criterion that can be used to evaluate the structural richness of an ontology is the **connection richness, ConnR**. It indicates the number of connected components of the concept graph, i.e., the number of trees. The root classes show the disconnected components, so for calculating *ConnR*, we determine the number of root classes. $ConnR(\mathcal{O}) = \frac{1}{No_root_classes}$. This metric can help if “islands” form in the knowledge based as a result of extracting data from separate sources that do not have common knowledge. The total **design metric richness (DMR)** is the combination (weighted sum) of these richness metrics.

- **Knowledge-base metric** This dimension describes the semantics and the content information of the ontology. We make use of several metrics: class richness, average population, and readability [60]. The **class richness, CR**, is an instance-based criterion used to reflect how instances in an ontology are distributed across classes. The class richness for an ontology \mathcal{O} can be defined as

follows: $CR(\mathcal{O}) = \frac{|C^I|}{|C|}$ where $|C^I|$ is the number of classes having instances. Another criterion that is important during the evaluation of the semantic richness of an ontology is the **descriptivity richness, DR**. This measure indicates the level of detail in the representation of the knowledge provided by the ontology. The descriptivity of an ontology can be defined as the number of concepts that have comments and/or labels. It can be defined as: $DR(\mathcal{O}) = \frac{|C^L|}{|C|}$ where $|C^L|$ is the number of concepts having comments and/or labels. The total **knowledge-base metric richness (KMR)** is the combination of these richness metrics.

- **Class metric** This dimension is used to reflect the relative importance of each concept in the ontology. To this end, we consider the context of a concept by including its superclasses, subclasses, and siblings. This metric is an important analysis metric depicting the concept sparseness in the ontology. This indicator can be used to select a suitable graph traversal algorithm during the partitioning process.

Given these three dimensions, we combine them to get the total richness of an ontology using a simple weighted-sum approach. Therefore, the **ontology richness (OR)** criterion can be defined as follows:

$$OR(\mathcal{O}) = w_1 \times DMR(\mathcal{O}) + w_2 \times KMR(\mathcal{O}) + w_3 \times CMR(\mathcal{O}) \tag{1}$$

where $DMR(\mathcal{O})$, $KMR(\mathcal{O})$, and $CMR(\mathcal{O})$ are the total design, knowledge-base, and class richness of the ontology (\mathcal{O}), respectively. w_1, w_2, w_3 are weights reflecting the importance of each of richness metric, such that $\sum_{i=1}^3 w_i = 1$. The normalized score is then listed for the user to decide to partition the ontology or look for another one.

Algorithm 1 Ontology modularization algorithm

```

Require: An ontology  $\mathcal{O}$ ,
Ensure: A set of module,  $\mathcal{M}(\mathcal{O}) = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K\}$ 
  { // Phase 1: Preprocessing }
1:  $\mathcal{M}(\mathcal{O}) \leftarrow \emptyset$ ;
2:  $\mathcal{CG} \leftarrow build\_ConceptGraph(\mathcal{O})$ ;
  { // Phase 2: Concept Ranking }
3: for  $c_i \in \mathcal{C}$  do
4:    $Score_c[i] \leftarrow Ranker(c_i)$ ;
5: end for
  { // Phase 3: Optimal number of modules determination }
6:  $\mathcal{CH} (= \{ \mathcal{CH}_1, \mathcal{CH}_2, \dots, \mathcal{CH}_K \}) \leftarrow$ 
    $determine\_Optimal\_No\_Modules(\mathcal{CG}, Score_c[])$ ;
  { // Phase 4: Partitioning }
7: initialize a partition with each  $\mathcal{CH}$  node;
8: add direct concepts of  $\mathcal{CH}$  to each corresponding partition;
9: for non-clustered  $c_i \in \mathcal{C}$  do
10:   $max\_sim \leftarrow 0$ ;
11:  for  $\mathcal{CH}_j \in \mathcal{CH}$  do
12:     $sim_{ij} \leftarrow MemFun(c_i, \mathcal{CH}_j)$ ;
13:    if  $sim_{ij} \geq max\_sim$  then
14:       $max\_sim \leftarrow sim_{ij}$ ;
15:       $concept\_place \leftarrow j$ ;
16:    end if
17:  end for
18:   $Clust.put(C_i, \mathcal{CH}_j)$ 
19: end for { // Phase 5: Module Generation }
20: for  $i \in \{1, 2, \dots, K\}$  do
21:   $\mathcal{M}_i \leftarrow Module\_generate(Clust.get(i))$ 
22:   $\mathcal{M}(\mathcal{O}).put(\mathcal{M}_i)$ 
23: end for

```

3.3 Modularization and Determination of Optimal Number of Modules Components

Once an ontology is investigated, the next step is to split the concepts \mathcal{C} of the concept graph \mathcal{G} into a set of disjoint partitions, and then generate a set of separate (possibly overlapping) modules $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$ such that the cohesion of concepts in one module is high, while the coupling of any two modules is low. To this end, we develop a seeding-based partitioning algorithm. The outline of the algorithm is described in the following, as shown in Algorithm 1. In the next section, we present a detailed description of the partitioning component.

- Ranking the concepts** The partitioning algorithm starts by selecting a set of nodes distinguished as *important nodes*, some of them are then elected to be cluster heads, \mathcal{CH} . In order to identify a node as an important one, we quantify its role in the concept graph. To this end, we introduce a new rank function, called *Ranker*, (Algorithm 1 lines 3–5). This function should be as simple as possible but effective, i.e., computing the *Ranker* function should not consume much time, but still correctly rank the concepts inside an ontology.
- Determining cluster heads** Once having computed the importance of the concepts of a concept graph, the next step is to select which concepts represent cluster heads, \mathcal{CH} . We have to cope with two questions: how many cluster heads should we select? and which cluster heads? If simply the nodes with the highest score were accepted as the cluster heads, the distribution of cluster heads across the concept graph would be disregarded. To avoid this problem, the distance between two cluster heads is measured, and among the highest scored nodes, those with at least a minimum distance of \mathcal{D} from each other are selected as the cluster heads. Furthermore, to estimate the optimal number of modules (cluster heads), we deploy an information theory-based approach to recommend this number, (Algorithm 1, line 6).
- Partitioning** The seed-based algorithm initiates one partition for each cluster head. Then, it places direct children in the corresponding cluster, and finally, for the remaining (non-clustered) nodes, we develop a membership function to assign remaining nodes to their fitting partition. The direct placement of children reduces the time complexity, since it reduces the number of comparisons by avoiding to compute the membership function for all concepts, (Algorithm 1, lines 7 – 19).
- Generating Modules** Once we have obtained the set of disjoint partitions (clusters), the following step is to generate a module for each partition preserving the required intra-relationships between concepts in the same partition as well as inter-links between concepts from different partitions, (Algorithm 1, lines 20 – 23).

In the following section, we provide more details on the ontology partitioning steps.

3.4 Evaluation Component

We need to evaluate and quantify the modularization process to make sure that the modularization solution meets the set of requirements. In general, the quality of an ontology (module) can be defined as the degree of conformance to functional and non-functional requirements [13,30]. This degree should be measurable. Current studies of the evaluation of modularization approaches focus on modularization algorithms and

the evaluation of the taxonomical structure of a created module [41]. According to [14], ontology evaluation determines the quality and adequacy of an ontology for reuse in a specific context for a specific goal.

To this end, in this paper, we make use of our ontology modularization evaluation metrics that can be used to assess the goodness of ontology modules [3]. To make this paper self-contained, we present some details about these metrics. In particular, we propose the module homogeneity (*HOMO*) as a metric of the internal characteristics of the set of concepts within the module, and the module heterogeneity (*HEMO*) as an assessment of interdependency between ontology modules.

4 SeeCont: The Ontology Partitioning Approach

In this section, we present technical details of the seed-based partitioning approach, called *SeeCont*. As shown in Fig. 1, *SeeCont* consists mainly of two components: the *modularization component*, and the *optimal number of modules component*. As mentioned, input ontologies are parsed and represented internally as *concept graphs*. We quantify the importance of graph concepts by introducing a new rank function exploiting the concept graph features. The number of cluster heads (\mathcal{CH}) is to be determined using the optimal number determination component. Finally, we assign the remaining graph concepts to their corresponding partitions (clusters) according to a proposed membership function. The outline of the *SeeCont* approach is shown in Algorithm 1 lines 5–23. In the following sections, we portray the description of each phase of the algorithm. To demonstrate the steps and procedure of the proposed approach, we use the *cmt* ontology⁴ illustrated in Figs. 2 and 3, where Fig. 2 represents the tree structure of the ontology (only the is-A relationships), while Fig. 3 represents the concept graph of the ontology. We select this ontology as an example for demonstration since it represents a very common domain (the conference domain), which will be easy to understand without the need for help from a domain expert. Furthermore, the *cmt* ontology has 29 concepts, but it has a quite enough number of relations (59 beside is-relations), which supports the description of the proposed approach.

4.1 Concepts Ranking

The partitioning algorithm starts by selecting a set of *important nodes*. Among them, a set of cluster heads, \mathcal{CH} , is identified. To this end, we developed the *Ranker* function.

⁴ an ontology from the conference track of OAEI (<https://oaei.ontologymatching.org/>).

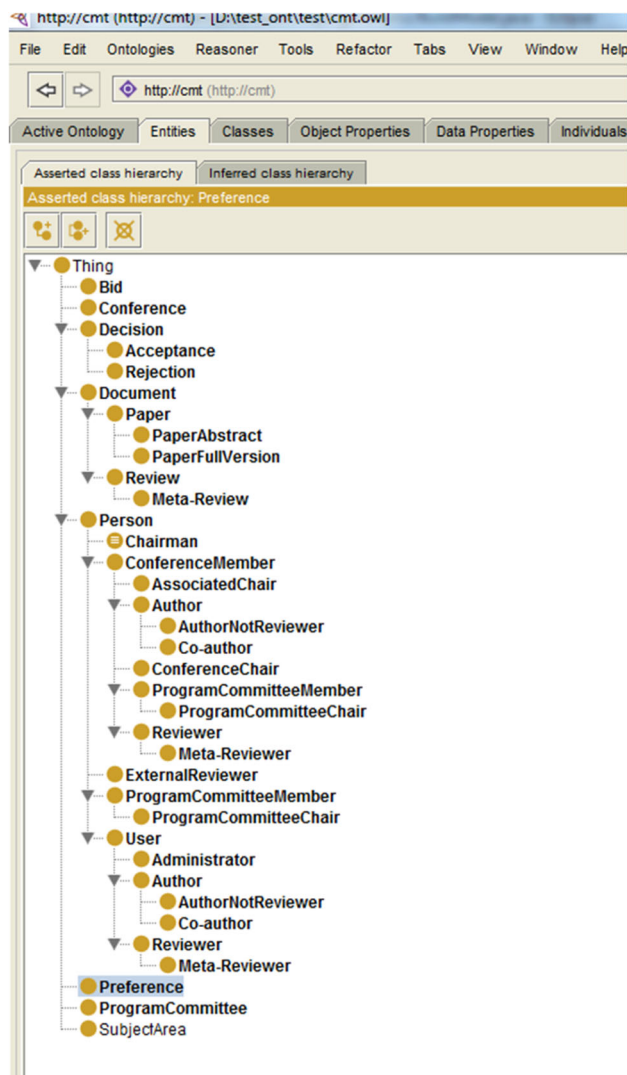


Fig. 2 Tree representation of “cmt”

The initial version of this function was based on the *centrality measure* of the concept. This centrality measure, derived from social network analysis [15], considers different aspects of centrality, such as *degree*, *closeness*, *betweenness*, and *stress*. Given that our problem is to deal with large-scale ontologies, despite its effectiveness, the centrality-based ranking function needs much time to rank the graph concepts.⁵ Therefore, we propose a new ranking function, which accounts for the context of the concept.

Concept context Given a *concept graph* $\mathcal{G} = (\mathcal{C}, \mathcal{R}, \mathcal{LAB})$, the context set of a concept $c_i \in \mathcal{C}$ is the set of surrounding concepts to a specific level d . The context includes the set of sub- and superclasses of c_i up to a level d . We also consider the set of properties a concept is involved in i.e.,

⁵ Interested readers can access our previous work [1].

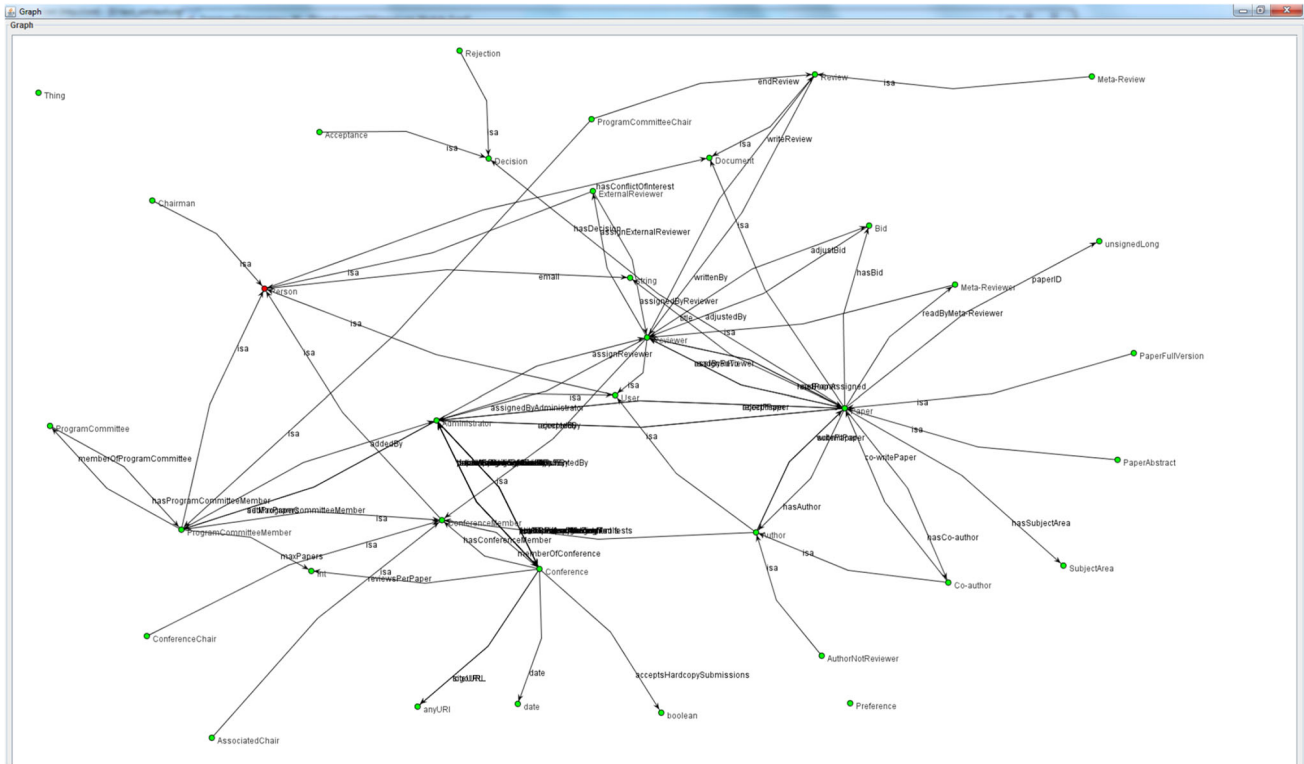


Fig. 3 Concept graph of “cmt”

$Contx(c_i, d) = \{SubClass(c_i, d) \cup SuperClass(c_i, d) \cup propSet(c_i)\}$, where $SubClass(c_i, d)$ and $SuperClass(c_i, d)$ are the subclasses and superclasses of c_i within d hierarchical levels, respectively, and $propSet(c_i)$ is the set of properties belonging to c_i . It is evident that the importance of a concept increases as it has a larger number of the entities in $Contx(c_i, d)$. In the current implementation, the value of the level d is determined based on the outcome of the analysis component.

Furthermore, in order to exploit more information a concept may have, we add the list of properties the concept has as well as other kinds of relations that connect the concept to other concepts. For example, the context of the concept “Paper” is the set of its superclasses *Document*, subclasses $\{PaperAbstract, PaperFullVersion\}$, its data properties $\{paperID, title\}$, and its object properties, such as $\{acceptPaper, acceptedBy, readPaper, rejectPaper, rejectedBy, hasAuthor, writePaper, \dots\}$. We can formulate the importance of a concept as follows:

$$\begin{aligned}
 Import(c_i) = & \omega_1 \times |SubClass(c_i, d)| \\
 & + \omega_2 \times |SuperClass(c_i, d)| \\
 & + \omega_3 \times |propSet(c_i)| + \omega_4 \times No_oth_rel
 \end{aligned}
 \tag{2}$$

where $|SubClass(c_i, d)|$, $|SuperClass(c_i, d)|$, $|propSet(c_i)|$, and No_oth_rel are the number of subclasses, the number of superclasses, the number of properties, and the number of other relations (e.g., equivalent) the concept c_i has to a level d . The weighting scheme is determined based on the results of the analysis phase, where $\sum_{i=1}^n \omega_i = 1$. Based on the computed importance value for each concept, we rank graph concepts.

Example 1 Applying the concept ranking method to the *cmt* ontology represented in Figs. 2 and 3 using a hierarchical distance $d = 2$, we get the result as shown in Table 1.

Table 1 shows that the *Person* concept is the most important concept using either tree-based or concept graph representation. However, the table gives two different rankings based on the type of information exploited during the ranking process. The table indicates that including more context results in better concept ranking. For example, the *Paper* concept does not appear when using the tree-based representation, while it has the second position when using the concept graph representation.

4.2 Model Selection and Clusters Head Determination

Once ranking the concepts, the next step is to decide how many concepts should be selected to constitute the cluster

Table 1 Top-5 concepts

Position	Using tree representation ($\omega_1 = 0.6, \omega_2 = 0.4, \omega_3 = \omega_4 = 0$)	Using concept graph ($\omega_1 = 0.3, \omega_2 = 0.3, \omega_3 = 0.3, \omega_4 = 0.1$)
1st	<i>Person</i>	<i>Person</i>
2nd	<i>ConferenceMember</i>	<i>Paper</i>
3rd	<i>User</i>	<i>Reviewer</i>
4th	<i>Document</i>	<i>ConferenceMember</i>
5th	<i>Author</i>	<i>Author</i>

heads (number of modules). Determining the proper number of modules for a given ontology is a really challenging problem due to the subjective nature of deciding what complements the correct partitioning. It can be considered as a trial-and-error process. Therefore, instead of using a single predefined number, \mathcal{K} , a set of values can be used. This set of values should satisfy some trade-off characteristics. First, it should be large enough to reflect a wide range of features of the input ontology. At the same time, it should be small enough to not consume much time and resources. To this end, we propose an information theoretical model to cope with these challenges. The proposed method makes use of the *Bayesian information criterion (BIC)* to estimate the optimal model. Here and within the context of ontology modularization, by a *model*, we mean the modularization output (the set \mathcal{K} of modules). Changing the value of \mathcal{K} , we get a new model. A optimal model is the modularization output with the optimal number of modules.

To achieve this goal, first, we estimate lower and upper bounds to reduce the search space of the optimization process. Then, we exploit BIC to evaluate the quality of each model generated from each iteration by introducing a new cost function based on the modules' properties.

4.2.1 Estimating the Interval

The search space of the optimization problem extends from a single cluster head to the size of the graph (the number of concepts). To reduce this search space, we propose an estimation for a range of values based on the characteristics of the concept graph, called the *boundary interval*. This range of values extends between a lower bound (\mathcal{LB}) and an upper bound (\mathcal{UB}). To achieve this goal, we introduce a definition for *point-wise*, (\mathcal{PW}) that captures information about the size of the graph as well as the hierarchy of concepts inside the ontology, as defined below.

$$\mathcal{PW} = \log(|\mathcal{C}|) \times \mathcal{AVD} \quad (3)$$

where $|\mathcal{C}|$ is the number of concepts in the ontology \mathcal{O} and \mathcal{AVD} is the average hierarchical distance of the concepts. The motivation behind using this formula is to combine some statistical information about the ontology such as the number of the concepts as well as structural information like the concept hierarchy. To compute the average hierarchical distance \mathcal{AVD} , we use the following formula that sums the concept hierarchy w.r.t. the total number of the concepts within the concept graph.

$$\mathcal{AVD} = \frac{1}{|\mathcal{C}|} \times \sum_{i=1}^{|\mathcal{C}|} \text{Path_length}(c_i) \quad (4)$$

where $\text{Path_length}(c_i)$ is the path length extending between the concept c_i and the root concept of the graph. After defining the point-wise \mathcal{PW} , we formulate the boundaries (\mathcal{LB} and \mathcal{UB}) as follows:

$$\begin{aligned} \mathcal{LB} &= \mathcal{PW} - \lambda \\ \mathcal{UB} &= \mathcal{PW} + \lambda \end{aligned} \quad (5)$$

while λ is an integer value added to give the proposed approach more flexibility during the boundaries estimation. The selection of λ is subjected to two trading-off factors. If we set larger values to it that means we cover a wide range of models, however, it will take much time to settle on a suitable model and vice versa. To this end, we set the values of λ between 5 and 10 (i.e., $5 \leq \lambda \leq 10$) depending on the number of concepts. If $\mathcal{LB} \leq 0$, we set it to 1 (as minimal number), while if $\mathcal{UB} > |\mathcal{C}|$, we set it to $|\mathcal{C}|$.

4.2.2 Model Selection

The next step is how to select the optimal model that represents the modularization output, i.e., how many modules should be generated by the partitioning process? To cope with this problem, we develop an information theoretic approach that makes use of the Bayesian information criterion (BIC) [55]. BIC is used to score individual models, and the model with the minimum BIC value can be chosen as the optimal model. A general description of the proposed approach is depicted in Algorithm 2.

The algorithm accepts an ontology \mathcal{O} to determine the optimal model that represents the modularization output. First, it identifies the boundaries for the model selection process by estimating the lower and upper bounds \mathcal{LB} and \mathcal{UB} , Algorithm 2 (line 1). It initializes \mathcal{K} with the lower bound value. The algorithm then iterates till \mathcal{K} reaches the upper bound value, (lines 3 – 8). In each iteration, we apply the *SeeCont* algorithm to get a candidate model, (line 4). This candidate model is then evaluated using BIC and the model with the minimum BIC value is recorded, (lines 5 and 6).

Algorithm 2 Optimal module numbers selection

Require: An Ontology, \mathcal{O}
Ensure: Optimal number of modules, \mathcal{K}
 1: $\mathcal{LB} \& \mathcal{UB} \leftarrow getLU_B(\mathcal{O});$
 2: $\mathcal{K} \leftarrow \mathcal{LB};$
 3: **while** $\mathcal{K} \leq \mathcal{UB}$ **do**
 4: $apply_SeeCont(\mathcal{K});$
 5: $BIC_{\mathcal{K}} \leftarrow computeBIC(Model_K);$
 6: $keep_minimum(BIC_{\mathcal{K}})$
 7: $\mathcal{K} \leftarrow \mathcal{K} + 1;$
 8: **end while**

4.2.3 Computing BIC

The problem of model selection can be stated as follows: Let \mathcal{O} be an ontology with n concepts. Based on the ontology characteristics, we get the value of the lower bound \mathcal{LB} and the upper bound \mathcal{UB} . Applying the *SeeCont* partitioning algorithm on the given ontology generates a set of models, $MOD_{\mathcal{LB}}, MOD_{\mathcal{LB}+1}, \dots, MOD_{\mathcal{K}}, \dots, MOD_{\mathcal{UB}}$. The arising question now is which model should be selected as the output of the modularization process. By a model MOD_i , we mean the set of modules generated from the partitioning process at specific run i , such that $\mathcal{LB} \leq i \leq \mathcal{UB}$. To settle on a precise model, we need to evaluate the quality of all candidate models. To this end, we make use of the BIC criterion as an evaluation metric for the model selection [52].

BIC is an approximation to Bayesian statistics and evidence [42]. It is precisely the quantity which updates the prior model probability to the posterior model probability. The Bayesian evidence, also known as the model likelihood, comes from the full implementation of Bayesian inference at the model level, which is very hard to calculate. Therefore, BIC as an approximation of the evidence that could be used, which is simpler but effective. It has been subsequently applied to the model selection problem, and we adopt it to the ontology modularization problem.

The BIC was introduced by Schwartz [52] and can be defined as:

$$BIC(MOD_i) = |\mathcal{C}| \times \ln(\mathcal{L}_{max}) + \mathcal{K} \times \ln(|\mathcal{C}|) \tag{6}$$

where $|\mathcal{C}|$ is the number of ontology concepts, \mathcal{K} is the number of modules, and \mathcal{L}_{max} is the maximum likelihood (ML) achieved by the model. We adopt the ML estimation since it has many elegant features. Among them are: *sufficiency* (complete information about the parameter of interest contained in its ML estimator); and *efficiency* (lowest-possible variance of parameter estimates) [37,39]. The problem can be stated as: given a set of observed data (set of concepts) and a model of interest (the set of modules) find a probability distribution function that is most likely to have produced the data. Under the assumption that when sets of observations are independent of one another and are normally distributed,

then maximizing the log-likelihood function is equivalent to minimizing the sum of square errors (*SSE*) [39]. Therefore, and since it is more simpler to compute in the context of ontology modularization, we make use of the *SSE* to represent the maximum likelihood for the ontology.

Given a model MOD_i containing \mathcal{K} of modules $M_{i1}, M_{i2}, \dots, M_{i\mathcal{K}}$ such that each module has a cluster head (\mathcal{CH}). We can consider the cluster head of each module as the central point of the module. In order to preserve the property that each module should contain the set of concepts with the minimum distance between them, we define the sum of square errors, *SSE*, for the model MOD_i as follows:

$$SSE(MOD_i) = \sum_{i=1}^{\mathcal{K}} \sum_{j=1}^{no_size} dist(\mathcal{CH}_i, c_j) \tag{7}$$

where *no_size* is the number of concepts inside the module M_{ij} , \mathcal{K} is the number of modules inside this model, and $dist(\mathcal{CH}_i, c_j)$ is the length of the minimal path between \mathcal{CH}_i and c_j . The reason behind the selection of this function is to reflect one of the criteria mentioned in Sect. 2 (*distance*) to ensure that an optimal partition is the partition where the set of its concepts having zero distance between each and the cluster head (\mathcal{CH}) of the partition.

4.2.4 Cluster Heads Selection

Once we obtain the optimal number of modules that an ontology can be partitioned into using the model selection component, the next step is to select the cluster heads among the set of ranked concepts. In order to have a good distribution of cluster heads over the concept graph, we should select concepts that are of distance \mathcal{D} from each other. For example, assume that the optimal number of modules for the *cmt* ontology shown in Figs. 2 and 3 is 3.⁶ Investigating the list of ranked concepts (see Table 1, the first concept (*Person*) will be directly selected as a cluster head. Then, we can select the second concept (*Paper*) as the next cluster head since it has no common parents/children with the already selected cluster heads. However, we do not choose the last cluster head from the remaining concepts in the list of Table 1 since all of them are direct children of the *Person* concept. We then follow the same procedure to select the *Review* concept as the third cluster head.

4.3 Finalizing Partitioning

At first, the *SeeCont* algorithm initiates one partition for each cluster head. Then, it places direct children in the corresponding cluster and finally, for remaining nodes, a membership

⁶ Actually the optimal number of modules for this ontology is 1, but we assume it 3 to use it during the explanation of the algorithm.

Table 2 Cluster after direct partitioning

1st partition	2nd partition	3rd partition
Person (\mathcal{CH}_1)	Paper (\mathcal{CH}_2)	Review (\mathcal{CH}_3)
ExternalReviewer	PaperFullVersion	Meta-Review
ProgramCommitteeMember	PaperAbstract	
ConferenceMember		
Chairman		
User		

function is used to determine the fitting partition of each node. In general, clustering is done through the following three steps:

- *Seeding* Creating a partition for each cluster head, *Algorithm 1*, (line 7).
- *Direct Spread* Assigning direct children of each cluster head to the corresponding partition, *Algorithm 1*, (line 8). By direct children, we mean that there is `is-A` relationship between a cluster head and the concept. For example, as shown in Fig. 2, the cluster head *Paper* has two direct children *PaperFullVersion* and *PaperAbstract* that can be assigned directly to this partition as shown in Table 2.
- *Indirect Spread* Calling a membership function for the remaining nodes, *Algorithm 1*, (lines 9–18).

Indeed, the *direct spread step* reduces the time complexity since the number of comparisons will be reduced as well as applying the membership function for all nodes. The result of applying the first two steps is presented in Table 2.

4.3.1 Membership Function

Once determining cluster heads (\mathcal{CH}) and assigning direct children to their proper heads, the next step is to place remaining concepts into their fitting partitions. To this end, we develop a membership function, *MemFun*, where each concept is associated with a flag, \mathcal{F} , such that if the \mathcal{F} of concept c is false, it means c is not assigned to any partition yet and thus, the membership function is called for the concept c . In addition, the \mathcal{F} flag can only be set once, i.e., each concept can be placed in only one cluster so that no overlap is observed in clusters. The membership function determines in which partition a concept $c_i \in \mathcal{C}$ should be placed. For this, the similarity of c_i with all \mathcal{CH} s is calculated and then c_i is placed in a cluster with the maximum similarity value. Using the proposed membership function, each concept is compared with cluster heads (\mathcal{CH} s), instead of comparing with all concepts like in [4,27], which also reduces the complexity of comparison.

In order to measure the membership of a concept to a cluster head, a linear weighted combination of the following

structural and semantic similarity measures is calculated as in the following equation:

$$MemFun(c_i, \mathcal{CH}_k) = \delta \times SNSim(c_i, \mathcal{CH}_k) + (1 - \delta) \times SemSim(c_i, \mathcal{CH}_k) \quad (8)$$

where δ is constant between 0 and 1 to reflect the importance of each similarity measure, *ShareNeighbors*(*SNSim*) and semantic similarity (*SemSim*) are two similarity measures that quantify the structural properties of the concept c_i , respectively.

4.3.2 Shared Neighbors

This measure considers the number of shared neighbors between c_i and \mathcal{CH}_k . The shared neighbor measure plays an important role in structural similarity, because similar concepts have similar neighbors [5,27]. The neighbors of a concept are the concept's children, concept's parents, concept's siblings, and the concept itself. In our implementation, we determine the neighbors of the concept c_i and the neighbors of the cluster head \mathcal{CH}_k , then determine how many concepts are common between these two sets.

$$SNSim(c_i, \mathcal{CH}_k) = \frac{|SN_{c_i} \cap SN_{\mathcal{CH}_k}|}{|SN_{c_i} \cup SN_{\mathcal{CH}_k}|} \quad (9)$$

where SN_{c_i} and $SN_{\mathcal{CH}_k}$ are the neighbor sets of the concept c_i and the cluster head \mathcal{CH}_k , respectively.

4.3.3 Hierarchical Semantic Similarity

It is evident that a higher semantic similarity implies a stronger semantic connection, so we calculate the semantic similarities between the concept c_i and the cluster head \mathcal{CH}_k . The most classic semantic similarity calculation is based on the concept hierarchy by identifying their lowest common ancestor [64]. To this end, for each concept, we extract local names of its surrounding (children, parents, and siblings) and make use of the I-sub similarity measure [57] to compute the semantic similarity between the concept c_i and the cluster head \mathcal{CH}_k . The hierarchy semantic similarity between c_i and

\mathcal{CH}_k can be defined as below:

$$\text{SemSim}(c_i, \mathcal{CH}_k) = \text{Comm}(c_i, \mathcal{CH}_k) - \text{Diff}(c_i, \mathcal{CH}_k) + \text{winkler}(c_i, \mathcal{CH}_k) \quad (10)$$

where $\text{Comm}(c_i, \mathcal{CH}_k)$, stands for the commonality between c_i and \mathcal{CH}_k , $\text{Diff}(c_i, \mathcal{CH}_k)$ for the difference, and $\text{winkler}(c_i, \mathcal{CH}_k)$ for the improvement of the result using the method introduced by Winkler. It should be that this similarity measure considers the commonalities of the two concepts as well as their differences during computing the similarity value.

During the partitioning phase, we consider the following points:

- The role of the analysis phase; After analyzing the candidate ontology, we get a set of metrics that describe the ontology. We make use of some of these metrics to adapt the partitioning process. For example, if the relation richness (RR) of the ontology is higher than a predefined value (in our implementation 0.1), then we should invest much time considering other kinds of relations than is-A during the computation of membership function. This explains why the concept *Reviewer* is assigned to the third partition not to the first one, as shown in Table 3.
- The second point that we should consider during the indirect assignment is the size of a cluster. Based on the criteria mentioned in Sect. 2, the set of resulting clusters should have a comparable size. For that, we have to trade-off between the quality of clustering solution and the cluster size. If we aim to have high quality solution, we do not need to set any constraint on the cluster size. But, in this case, different clusters will have un-comparable sizes. On the other hand, to have such clusters with comparable size, we have to set a condition on each cluster size. For example, the concept *ConferenceChair* is closer to the cluster head *Person* than the cluster head *Paper*, but since we are applying the cluster size limit, then the concept *ConferenceChair* is assigned to the second partition not to the first one, as shown in Table 3.
- Equal similarity values; when computing the similarity between a concept and the given set of cluster heads, it happens that two or more cluster heads have equal similarity values to the same concept. In that case, a more sophisticated similarity measure is used to resolve this equal value issue.

4.4 Generating Modules

Once we have the set of disjoint partitions, each contains a set of concepts as shown in Table 3, the next step is to construct a set of consistent modules corresponding to the set

of partitions in the presence of the original ontology. Each module should represent a stand-alone ontology, which can be later re-used. To this end, we reconstruct each module to preserve the full knowledge from the original ontology. During the construction process, we keep inter-module links to be able to reconstruct the original ontology, given the output set of modules, as shown in Fig. 4. The figure shows that this module contains the set of concepts from the corresponding partition, *Review*, *MetaReview*, *Reviewer*, *MetaReviewer* as well as another set of concepts, which are necessary to constitute a consistent and self-contained ontology.

5 Evaluation

In order to evaluate our proposed system, we looked at two different aspects: First, we ran a number of experiments using a set of ontologies with different characteristics and measured a number of performance metrics. This part of the evaluation shows that our approach is sufficiently efficient to be of use and that the metrics we define are indeed achieved. Second, we performed a user survey to evaluate whether the method delivers a partitioning that meets user expectation that is whether the metrics we defined to indeed result in meaningful partitions from a user perspective. This evaluation is thus focused on the effectiveness of the approach. To evaluate the performance of the proposed approach, we conducted a set of experiments utilizing a set of ontologies. We ran all our experiments on a 3.4GHz Intel (R) Core i7 processor with 16GB RAM running Windows 7. The proposed approach has been developed and implemented in Java. The tool implementation is currently available through GitHub under the following link (<https://github.com/fusion-jena/OAPT>). In the following evaluation, we aim to validate the quality of the *OAPT* components, particularly, the analysis component and the modularization component.

5.1 Dataset

We validated the proposed approach using several ontologies (33 ontologies) collected from different domains, (such as biological, medical, health, environmental, and generic domains) and having different characteristics, as shown in Table 4. Some of these ontologies and their characteristics have been collected from BioPortal⁷ and some others have been collected from the ontology matching evaluation.⁸

⁷ <https://bioportal.bioontology.org/>.

⁸ <https://oaei.ontologymatching.org/>.

Table 3 Cluster after final partitioning

Step	1st partition	2nd partition	3rd partition
Initial	<i>Person (CH₁)</i>	<i>Paper (CH₂)</i>	<i>Review (CH₃)</i>
Direct	<i>ExternalReviewer</i> <i>ProgramCommitteeMember</i> <i>ConferenceMember</i> <i>Chairman</i> <i>User</i>	<i>PaperFullVersion</i> <i>PaperAbstract</i>	<i>Meta-Review</i>
Indirect	<i>Conference</i> <i>ProgramCommittee</i> <i>ConferenceMember</i> <i>Preference</i> <i>Administrator</i> <i>AuthorNotReviewer</i> <i>Author</i> <i>Co-author</i> <i>ProgramCommitteeChair</i>	<i>Bid</i> <i>Decision</i> <i>Document</i> <i>Rejection</i> <i>Acceptance</i> <i>AssociatedChair</i> <i>ConferenceChair</i> <i>SubjectArea</i>	<i>Reviewer</i> <i>Meta-Reviewer</i>

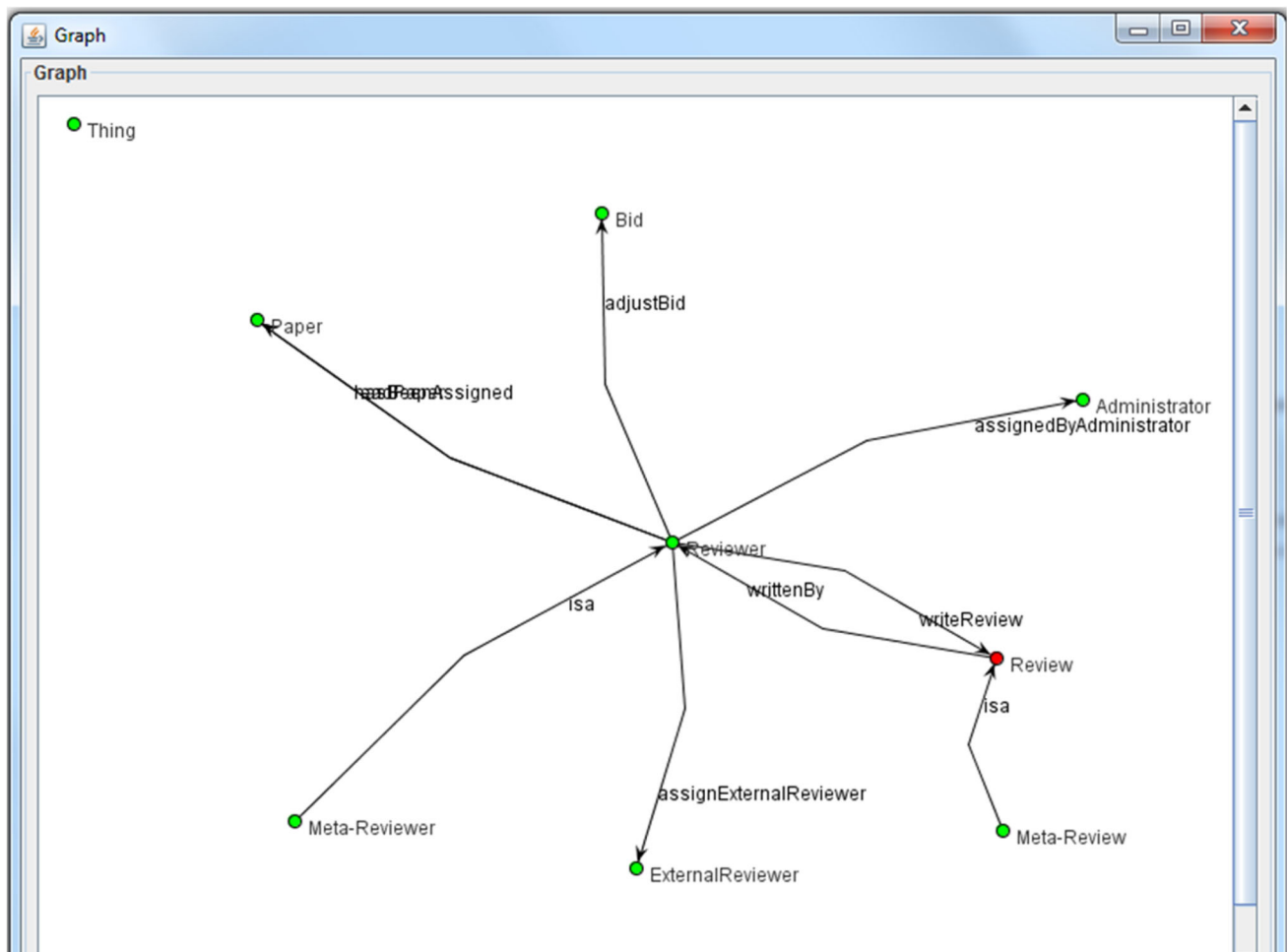


Fig. 4 Module_3 of “cmt”

Table 4 Ontology data set

Ontology	Domain	No. class	Ontology	Domain	No. class
ADO	Medical (Alzheimer's disease ontology)	1564	AIDSClinic	Medical	159
ASDPTO	Phenotype(Autism Spectrum Disorder Phenotype Ontology)	284	Asthma	Health	209
BCO	Biological Collections Ontology	126	BFO	Basic (Basic Formal Ontology)	35
BHN	Health (Biologie Hors Nomenclature)	2534	BMT	Medical (Biomedical Topics)	252
BP	Biological (BioPAX Ontology of Biological Pathways)	68	CBO	biological (Cell Behavior Ontology)	241
CCON	Plant (Cerrado concepts and plant community dynamics)	85	CHEBI	Chemical Entities of Biological Interest Ontology	45,436
CMT	conference	40	CN	Computer Network	537
Conference	conference	92	ENVO	Biodiversity (Environment Ontology)	6191
EOL	Environment Ontology for Livestock	648	EPILONT	Biomedical (Epilepsy Ontology)	137
FLOPO	Plant (Flora Phenotype Ontology)	26,866	GFO-basic	General (General Formal Ontology)	44
GFO-bio	Organisms, Anatomy	166	mouse_anatomy	Anatomy	2743
NCL_anatomy	Anatomy	3304	OBOE-sbc	Observation (Extensible Observation Ontology)	630
Ontobio	Biodiversity (Biodiversity Ontology)	188	OntoDM	Data mining	1501 (unnamed classes)
PATO	Phenotype (Phenotypic Quality Ontology)	2603	PL	philosophical	377
PW	Biological (Pathway Ontology)	1852	ROO	Health (Radiation Oncology Ontology)	1182
SBO	Biology (Systems Biology Ontology)	640	TOK	Terminological and Ontological Knowledge Resources Ontology	192
Travel	Travel	34	Travel	Travel	34

5.2 Experimental Results

The main focus of the *OAPT* tool is to analyze and partition ontologies, thus, we demonstrate the effectiveness of the tool w.r.t. each component first and then discuss the overall performance of the tool.

5.2.1 Ontology Analysis

To validate the performance of the ontology analysis component, we conducted a set of experiments utilizing the set of ontologies listed in Table 4. We first extracted ontology basic

metrics for each ontology, and we then computed the ontology richness based on these metrics. The results are reported in Tables 5 and 6.

Table 5 illustrates a set of basic ontology metrics, such as the number of ontology named classes (*No. class*), the number of total classes (*No. of total class*), the number of object and data properties (*No. of object prop.*, *No. of data prop.*), etc. as well as the time needed to extract these metrics and carrying out the ontology analysis. The table shows that there are few ontologies without blank nodes, i.e., the num-

⁹ We get these metrics without applying any reasoner

Table 5 Ontology basic metrics

Ontology	No. class	No. of total class	No. of sub	No. of object prop	No. of data prop	No. of individual	No. of comments	No. of labels	time (sec)
ADO	1564	2338	1594	12	0	0	812	1560	1.0
AIDSClinic	9	159	2	0	0	0	4	9	7.809
ASDPTO	284	284	283	0	0	0	0	284	0.074
Asthma	209	209	203	0	0	0	0	0	0.124
BCO	126	207	126	44	159	18	48	119	0.28
BFO	35	35	34	0	0	0	0	35	0.047
BHN	2534	2534	2542	0	0	0	0	0	0.444
BMT	252	252	272	12	0	0	52	252	0.081
BP	68	147	68	55	41	0	67	0	0.051
CBO	241	241	239	25	16	0	1	240	0.061
CCON	85	249	81	28	0	83	82	85	0.085
CHEBI	45,436	94,484	71,828	9	0	0	0	45,436	18.296
CMT	29	40	24	49	10	0	3	0	0.054
CN	537	1165	556	38	0	2	78	0	0.113
Conference	59	92	46	46	18	0	0	0	0.051
ENVO	6191	8134	8281	115	0	0	354	6191	2.4
EOL	648	650	658	1	0	0	0	648	0.138
EPILONT	137	137	133	5	5	0	52	136	0.025
FLOPO	26,866	99,762	35,348	17	0	0	917	26,866	12.539
GFO-basic	44	76	44	41	0	0	1	44	0.056
GFO-bio	166	262	179	73	0	1	47	0	0.168
Mouse_anatomy	2743	4374	1807	3	0	345	0	2743	0.964
NCI_anatomy	3304	4966	3761	2	0	6104	0	3304	0.705
OBOE-sbc	630	1647	635	26	6	0	102	227	7.392
Ontobio	188	221	173	58	43	267	0	0	2.03
OntoDM	0	1501	0	90	0	0	0	0	5.071
PATO	2603	3253	1852	23	0	0	91	2519	0.58
PL	377	992	407	313	0	0	377	0	0.309
PW	1852	1972	2211	1	0	0	16	1843	0.601
ROO	1182	1733	972	208	2	0	0	1182	0.216
SBO	640	640	691	0	0	0	639	640	0.118
TOK	192	566	178	95	18	25	28	72	0.109
Travel	34	60	24	6	4	14	8	0	0.01

Table 6 Ontology analysis metrics

Ontology	w.r.t. named classes				w.r.t. total classes			
	Design metric	KB metric	class metric	richness	Design metric	KB metric	class metric	richness
AIDSClinic	0.4	0.33	0.54	0.42	0.4	0.03	0.01	0.15
ADO	0.4	0.33	0.27	0.33	0.4	0.33	0.34	0.36
ASDPTO	0.4	0.33	0.52	0.42	0.4	0.33	0.52	0.42
Asthma	0.4	0	0.52	0.31	0.4	0	0.52	0.31
BCO	0.53	0.38	0.22	0.38	0.53	0.25	0.31	0.36
BFO	0.4	0.33	0.62	0.45	0.4	0.33	0.62	0.45
BHN	0.4	0	0.52	0.31	0.4	0	0.52	0.31
BMT	0.4	0.33	0.51	0.41	0.4	0.33	0.51	0.41
BP	0.47	0.33	0.19	0.33	0.44	0.15	0.24	0.28
CBO	0.41	0.33	0.52	0.42	0.41	0.33	0.52	0.42
CCON	0.37	0.66	0.55	0.53	0.38	0.33	0.16	0.29
CHEBI	0.4	0.33	0.04	0.26	0.4	0.16	0.24	0.27
CMT	0.41	0.03	0.33	0.26	0.41	0.03	0.35	0.26
CN	0.39	0.05	0.1	0.18	0.4	0.02	0.23	0.22
Conference	0.43	0	0.18	0.2	0.43	0	0.28	0.24
ENVO	0.4	0.33	0.34	0.36	0.4	0.27	0.38	0.35
EOL	0.4	0.33	0.51	0.41	0.4	0.33	0.51	0.41
EPILONT	0.41	0.33	0.54	0.43	0.41	0.33	0.54	0.43
FLOPO	0.4	0.33	0.5	0.41	0.4	0.09	0.13	0.21
GFO-basic	0.36	0.33	0.22	0.3	0.36	0.2	0.31	0.29
GFO-bio	0.38	0.1	0.25	0.24	0.38	0.06	0.33	0.26
mouse_anatomy	0.4	0.38	0.21	0.33	0.4	0.24	0.31	0.32
NCL_anatomy	0.4	0.67	0.25	0.44	0.4	0.56	0.33	0.43
OBOE-sbc	0.4	0.18	0.35	0.31	0.4	0.07	0.19	0.22
Ontobio	0.43	0.34	0.41	0.39	0.43	0.34	0.42	0.4
OntoDM	0	0	0	0	0	0	0	0
PATO	0.4	0.33	0.19	0.31	0.4	0.27	0.25	0.31
PL	0.36	0.33	0.34	0.34	0.38	0.13	0.19	0.23
PW	0.4	0.33	0.47	0.4	0.4	0.31	0.47	0.39
ROO	0.38	0.33	0.27	0.33	0.39	0.23	0.34	0.32
SBO	0.4	0.33	0.51	0.41	0.4	0.33	0.51	0.41
TOK	0.39	0.22	0.51	0.37	0.39	0.07	0.17	0.21
Travel	0.42	0.22	0.12	0.25	0.42	0.12	0.21	0.25

ber of named classes is equal to the number of total classes, e.g., the *ASDPTO* and *SBO* ontologies. This kind of ontologies needs less time to carry out the analysis process. The table also indicates that ontologies containing higher number of blank nodes compared to the number of named nodes require more time to be analyzed. For example, the *AID-SClinic* ontology has 159 total classes, only 9 of those are named classes, however, the tool needs more than 7 seconds to analyze this ontology. Another important finding from this table is that ontology engineers are not engaging with documenting and annotating ontology concepts, which makes it hard to reuse them. For example, only 12 (out of 33) ontolo-

gies have reasonable amount of comments and annotations (15% of the named classes have comments). Furthermore, the table also shows that only one-third of these ontologies do not have any label.

Based on these basic metrics, we computed a set of analysis metrics including the ontology design metric (*design*), the knowledge base metric (*KB*), the class metric (*class metric*), as well as the ontology richness (*richness*) as a combined metric for these individual metrics, as given in Table 6. Since these analysis metrics are computed based on basic metrics, we consider two cases: the first is using the number of named classes, while the second is considering the total number

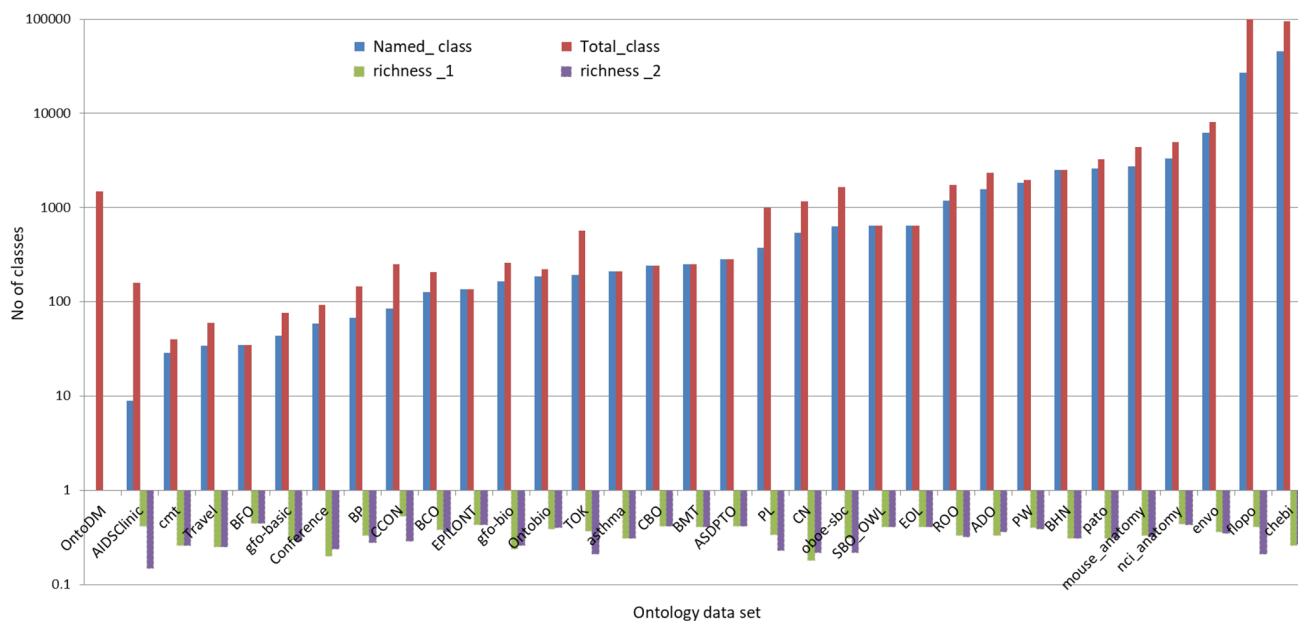


Fig. 5 Ontology richness w.r.t number of named classes

of classes during computing these metrics. The results are reported in Table 6. In general, ontologies having small difference between the number of named classes and the number of total classes such as *ASDPTO*, *Asthma*, *EOL*, and *SBO* have approximately the same *richness*. However, ontologies with big differences such as *FLOPO*, *AIDSClinic* and *OBOE-sbc* have different *richness*. We also observed that ontologies with high numbers of object and data properties have higher *design metric* values. For example, the *CMT* ontology has 59 object and data properties, while it has only 24 *is-a* (subclass) relations. This is a very important indicator when partitioning such ontologies. It tells us we have to focus more on relations other than the *is-a* relations. Another example is the *mouse_anatomy* contains only 3 object properties and 1807 *is-a* relations. So, it is a simple hierarchy, and we need to focus only on these simple relations.

Table 6 also illustrates the distribution of semantic-related information, such as individuals and comments within ontology through the *KB metric*. For example, the *Asthma*, *OntoDM* and *Conference* ontologies have no such semantic information, so their *KB metric* values are 0. However, the *NCI_anatomy* ontology has the highest *KB metric* since it has more such semantic information. As shown in Table 5, it has a large number of individuals and labels, but it has no comment. So, its *KB metric* value is less than one. Table 6 as well shows that the two *KB metric* values for the same ontology are not the same, if the number of named classes is less than the total number of classes. This is also the same observation when considering the ontology *class metric*. Another interesting observation from the table is that the *AIDClinic* ontology has a high *class metric* value (in the case of consid-

Table 7 OntoComplex results

Ontology	Ontology-level				Class-level			
	SOV	ENR	TIP	EOG	NOC	DIT	CID	COD
<i>CMT</i>	89	0.787	11	2.9	0.86	2.3	1.06	2.34
<i>BCO</i>	347	0.936	41	2.55	1.15	4.52	1.96	1.47

ering only named classes). This can be explained as follows: this ontology has only 9 named classes and 2 subclass relations, as shown in Table 5. So, the relative importance of a class with the ontology is high, which increases its *class metric* value. But, this value decreases when considering the total number of classes, since the relative importance of a class becomes very small. The *class metric* is an important indicator for the ontology reuse, especially our context “ontology modularization.” Such that a high class metric means that the relative importance of a class in the ontology is high, and there is strong connection between ontology concepts. So, the number of modules generated by partitioning this ontology should be small.

To summarize these observations, we figure out the relationship between the number of classes and the ontology richness, as depicted in Fig. 5. In general, the two ontology richness values (*richness_1* and *richness_2*) are the same if the number of named classes (*No. class*) is equal to the total number of classes (*No. of total classes*), such as *Asthma* and *SBO* ontologies. However, there is an exception for that rule. The *OntoDM* ontology has the same ontology richness value for the two different cases. This ontology has no named classes, it only has a set of object properties, which affects

Table 8 OntoMetric results

Ontology	schema		Knowledge		class NOC	-	graph	
	Rel. rich	Class rel.	Avg. pop	Class rich.			Av. depth	Av. breadth
<i>CMT</i>	0.7	0.27	0	0	1.67	-	1.06	2.34
<i>BCO</i>	0.55	0.28	0.21	0.03	4.52	-	5.5	3.07

Table 9 Concept ranking for the *CMT* ontology

rank	$d = 1$	$d = 2$	$d = 3$
1st	ConferenceMember	Person	Person
2nd	Person	ConferenceMember	ConferenceMember
3rd	Author	User	User
4th	User	Author	Author
5th	Paper	Reviewer	Reviewer
6th	Reviewer	Document	Document
7th	Decision	Paper	Paper
8th	Document	ProgramCommitteeMember	ProgramCommitteeMember

on *design metric* value. One more interesting finding is that the *FLOPO* richness values are highly different due to the big difference between the number of named classes and the total number of classes the *FLOPO* ontology have.

Ontology analysis comparison Although the importance of ontology evaluation within different application scenarios, a quite few number of tools and metrics have been proposed and developed to validate the quality of ontologies, such as *OntoMetrics*¹⁰[34], *OntComplexity*¹¹ [66], and *Ontology Auditor* [9]. We select the first two metrics to compare with for their availability. *OntComplexity* proposes a set of metrics based on the graph-centric representation of ontologies. This set of metrics is grouped into two categories: *Ontology-level* and *class-level* metrics. The size of vocabulary (SOV), the edge node ratio (ENR), tree impurity (TIP), and entropy of graph (EOG) are ontology-level metrics, while number of children (NOC), depth of inheritance (DIT), class-in-degree (CID), and class-out-degree (DOG) are class-level metrics. The *OntoMetric* includes a set of predefined metrics, which can be classified into four categories: *schema* and *graph* from [16], while *knowledgebase* and *class* metrics from [60]. For the comparison, we use *cmt* and *BCO* ontologies. The results are reported in Tables 7 and 8.

Table 7 shows the results of applying *OntComplexity*, while Table 8 represents results of *OntoMetrics*. Table 7 illustrates that the *BCO* ontology has a higher edge-to-node ratio than the *CMT* ontology, which indicates that *BCO* has a higher connectivity density. The table also indicates that the *BCO* ontology has a less EOG value than the *CMT* ontology, which indicates the existence of more structural patterns

Table 10 Ranking time for *CHEBI* concepts

d	time (sec)	memory (MB)
1	452.2	30.0
2	850	70.4
3	1131.5	135

and *BCO* is more regular and less complex than *CMT*. These results are consistent with our results illustrated in Table 6, which indicate *BCO* has a higher design metric than *CMT*. Also, comparing our analysis results to the results obtained by *OntoMetrics* (Table 8) demonstrates that our results are consistent with *OntoMetrics*.

5.2.2 Ontology Modularization

In this section, we validate the performance of ontology modularization components including: *ranking*, *model section*, and *partitioning*.

Concept Ranking To quantify the importance of concepts inside a concept graph, we conducted a set of experiments to examine the effect of the level on the concept ranking. We apply the ranking algorithm on the *CMT* ontology¹² with different levels (different values of d that appears in the concept context definition). The results are reported in Table 9. The table shows that as more concept contexts we consider, as we determine the correct importance of the concept. For example, when we use the set of subclasses and superclasses as the concept context, i.e., $d = 1$, we have the *conferenceMember* concept as the top-1 concept. However, considering more

¹⁰ <https://ontometrics.informatik.uni-rostock.de/ontologymetrics/>.

¹¹ <https://users.monash.edu/~yli/ontComplexity/index.html>.

¹² We selected it for the purpose of presentation, since it is easily to present results with an ontology with 30 concepts.

surrounding concepts, increasing the value of d , the *User* concept becomes the top-1 concept. However, this requires a penalty to be paid as the time needed to consider such surroundings. In such small ontologies, it does not matter, but for large-scale ontologies, we did another test to study the effect of d on the ranking time. For this, we carried out this experiment using the *CHEBI* ontology. The results are reported in Table 10. The table shows that as more contexts the ranking algorithm considers, the more resources it requires. Therefore, in our implementation, we trade-off between the ranking quality with the minimum resources required, and we select the ranking process with the second level, i.e., $d = 2$.

Model selection In our implementation, we attempt to get answers to the following questions:

- What is the added value behind using interval boundaries (\mathcal{LB} and \mathcal{UB})?
- Which λ value should be used during estimation \mathcal{LB} and \mathcal{UB}

To answer the first question, we used a set of ontologies, listed in Table 4 and carried out two sets of experiments: the first using the interval boundaries (\mathcal{LB} and \mathcal{UB}) and the second set without involving them. We measured both the generated “optimal” number of modules and the time needed to finish the modularization task. The results are reported in Figs. 6 and 7. The first figure shows that using the boundary interval generates smaller number of modules than the case when no boundary interval has been used. Figure 6 also shows that small ontologies, like *CMT*, *Conference*, and *GFO-basic* ontologies, have been modularized into only one module using both cases, however, it takes more time to carry out the modularization process with using the boundaries than without using the boundaries, as shown in Fig. 7. But in general, as illustrated in Fig. 7, partitioning larger ontologies involving the boundary intervals requires less times compared to partitioning these ontologies without involving the intervals. Figures 6 and 7 also demonstrate that using the boundary interval outperforms the case where no interval has been used w.r.t. the time needed to generate these optimal models. For example, modularizing the *BCO* ontology using the interval boundary into 4 modules needs only 0.45 seconds to complete the modularization process, however, without using the boundary interval, it has been modularized into 32 modules in 10 seconds. It should be noted that using the interval boundary enables the approach to modularize bigger ontologies, such as the *FLOPO* Ontology with 26,866 named classes needs 600 seconds to estimate its optimal number of modules (39), while it needs 18 seconds to estimate the optimal number of modules for the *PATO* ontology with 3253 concepts. For the *CHEBI* ontology with 102,124 concepts, the partitioning algorithm needs about 16 minutes to

run the partitioning process using the boundaries, however, it needs more than two days, so we stop it before completing the modularization without using the boundary interval. So, this is missing as shown in Fig. 6. This demonstrates the effectiveness of using the lower and upper bounds within the modularization process.

To validate the value of λ selecting during our experiments, we carried out another set of experiments using a subset of ontologies listed in Table 4. For each ontology, we run each experiment ten times for different values of λ . The results are reported in Fig. 8. The figure shows that varying the value of λ has a great impact on the estimated value of optimal number of modules, especially for larger ontologies. Furthermore, increasing the values of lambda ($\lambda \geq 5$), we found that the lower bound especially for smaller ontologies becomes always one. Therefore, we decide to select lambda equal to 5 in our experiments.

Partitioning In this section, we carried out another set of experiments to validate the performance of the partitioning approach w.r.t. the set of criteria mentioned in Sec. 2 and described also in [41]. These set of criteria evaluate several perspectives of the approach: *module quality*, *tool performance*, and *usability*.

Module Quality

- **Cohesion and Coupling** To consider these criteria in our evaluation, we make use of our new criteria used to evaluate ontology modularization [3]. We consider the *intra-module* similarity as a measure for the module cohesion (module homogeneity, *HOMO*), and the *inter-module* similarity as a measure for the module coupling (module heterogeneity, *HEMO*). By intra-module similarity, we mean the similarity between concepts within the same module, and it should be high to reflect the module homogeneity. On the other hand, the inter-module similarity captures the similarity between different modules generated from partitioning, and it should be small to reflect inter-module heterogeneity. We carried out a set of experiments to validate these criteria taking into account both with interval and without interval cases making use of the dataset presented in Table 4. The results are reported in Figs. 9 and 10, where Fig. 9 shows the performance of the tool w.r.t the intra-module similarity (*intra-sim*) and Fig. 10 depicts the inter-module similarity performance. Generally speaking, these figures demonstrate that the partitioning approach is effective using both with_interval and without_interval cases. For the given data set, it achieves an average *intra-module* similarity of 0.36 (0.328) using without_interval

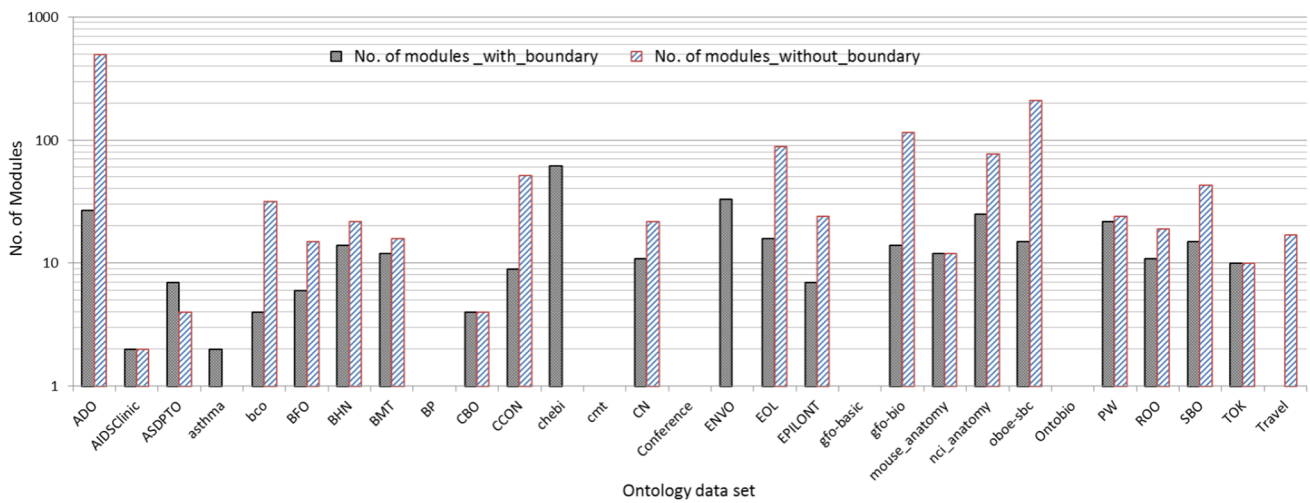


Fig. 6 Optimal number of modules, K

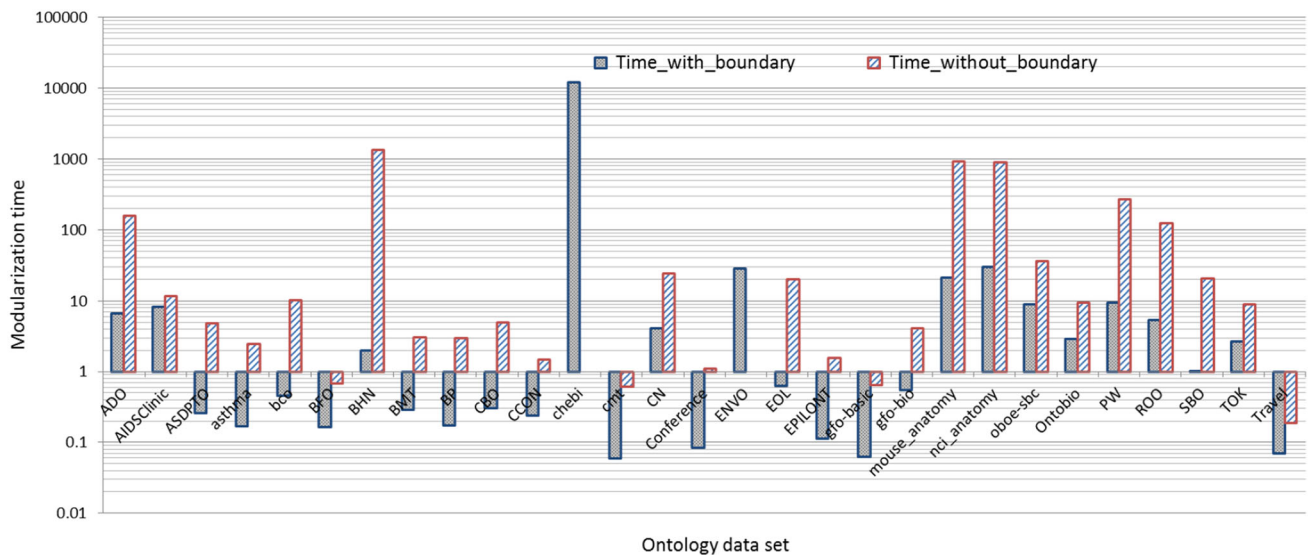


Fig. 7 Modularization time, sec

and using `with_interval`, respectively. It also attains an average inter-module similarity of 0.155 (0.064) using `without_interval` and `with_interval`, respectively. One more important finding extracted from this set of experiments is that partitioning ontologies without estimating interval boundaries results in more modules than using the estimated interval boundaries. For the given data, the proposed tool generates a total of 346 and 1317 modules in the case of using `with_boundary` and `without_boundary`, respectively. This generates modules with fewer concepts compared to generated modules using the lower and upper bounds. Therefore, the intra-module similarity using `without_interval` ($intra_sim_without$) is higher than using `with_interval` ($intra_sim_with$). However, the difference between the averages of

these two values is not so large, as shown in Fig. 9 ($\frac{av_intra_sim_without}{av_intra_sim_with} = \frac{0.36}{0.328} = 1.09$). On the other hand, partitioning an ontology using lower and upper bounds outperforms partitioning without these bounds for the following reasons: it allows partitioning large ontologies, such as *CHEBI* as shown in Figs. 6 and 7; it requires less time to carry out the partitioning; it produces more loosely coupled modules, as shown in Fig. 9 ($\frac{av_inter_sim_without}{av_inter_sim_with} = \frac{0.155}{0.064} \approx 2.41$); and it preserves almost the same cohesion.

- **Size** In this evaluation, we consider the size of a module as the number of concepts in the module. The aim of this evaluation is to validate the importance of module size on the quality of modularization. In fact, it is too hard to rely on the absolute module size as an evaluation criterion. To this end, we consider two

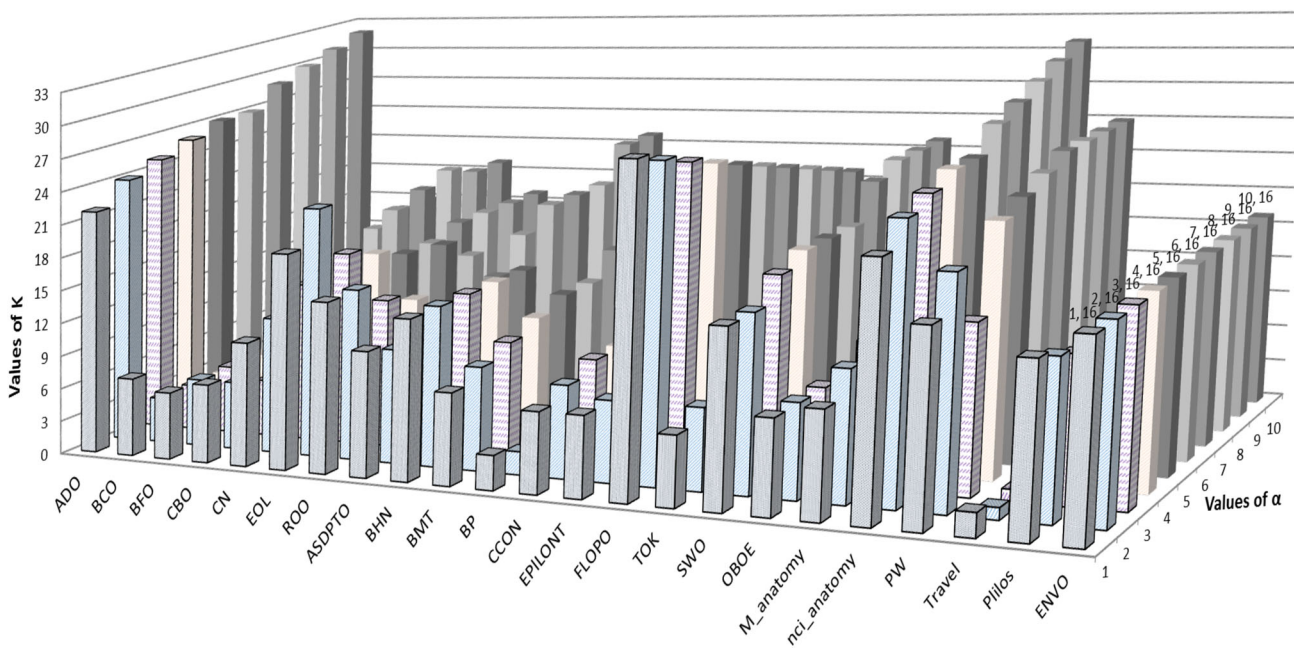


Fig. 8 Studying effect of λ on K

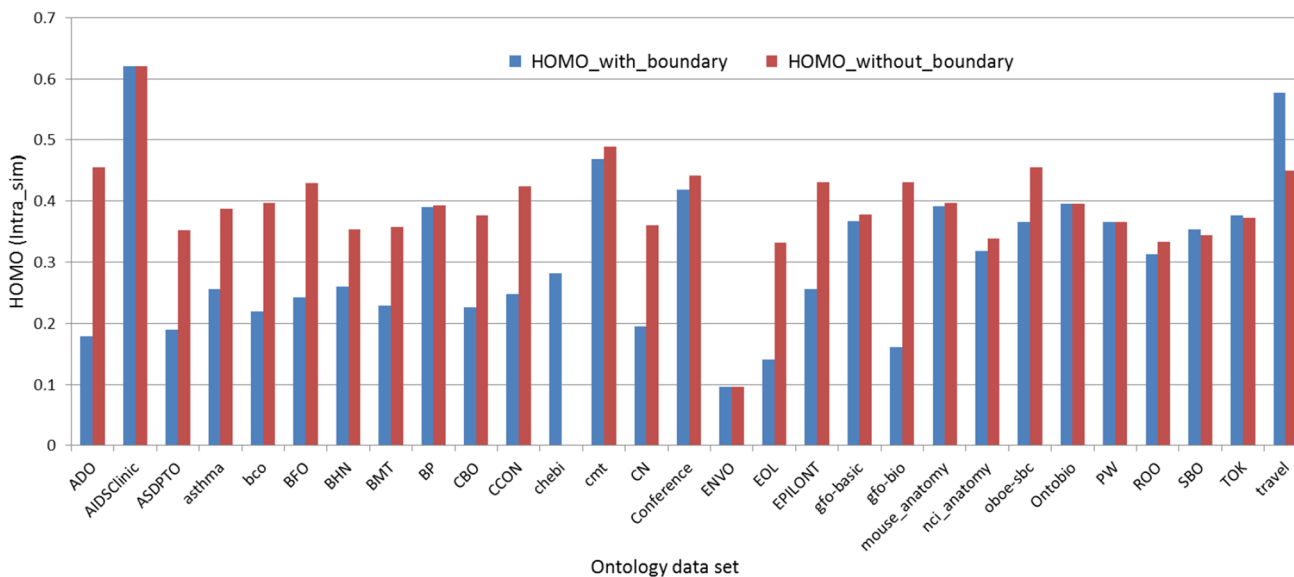


Fig. 9 Intra-module similarity (HOMO)

different scenarios: (i) the first is to examine the ratio between the number of concepts belonging to modules with the maximum and the minimum number of concepts. Let $Size_Min$ and $Size_Max$ be the module size of modules with the minimum and the maximum number of concepts, respectively. Then, the size ratio is defined as $ratio = \frac{Size_Min}{Size_Max}$. (ii) The second issue is to compute the relative size (RS) among all modules. We define the average relative size as follows: $RS = \sum_{i=0}^{k-1} \sum_{j=i+1}^k \frac{|M_i| - |M_j|}{size}$. By this measure,

we aim to ensure that the concepts of the original ontology have been distributed among all generated modules. We then study the effect of module size on both the cohesion and the coupling of the modularization. The results for the first test of experiments are shown in Table 11, which are ordered by the number of generated modules.

The table shows that ontologies with only one module have a ratio of 1 and a relative sizes of 0, which

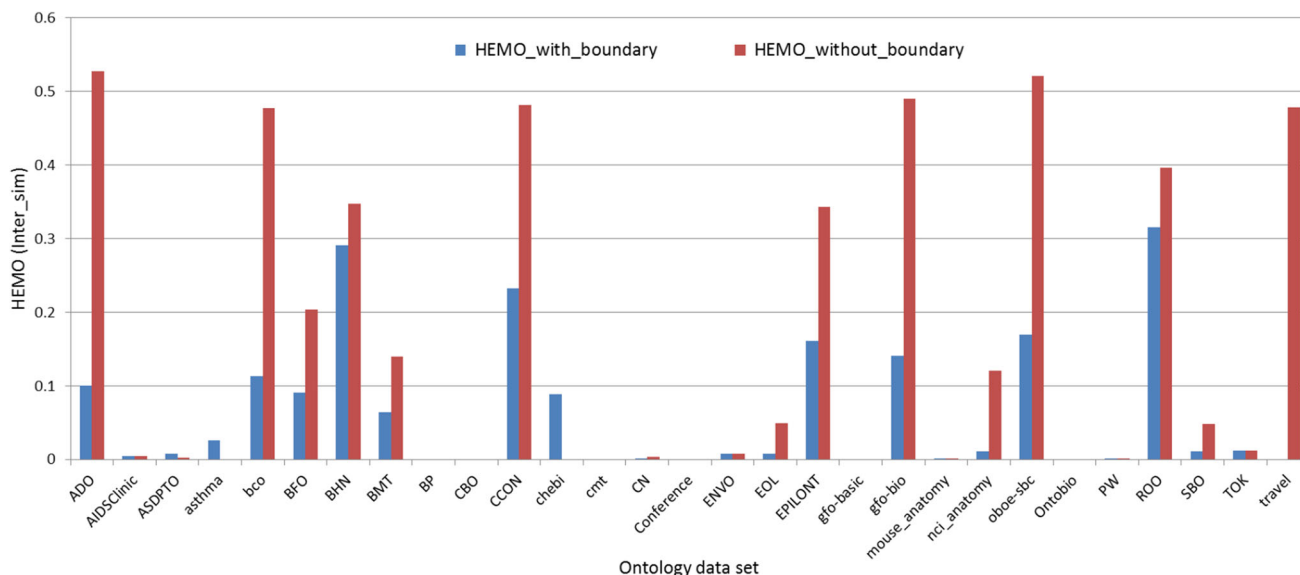


Fig. 10 Inter-module similarity (HEMO)

is a natural property. The table also shows that ratio values decrease (in general with a few exceptions) as the number of generated modules increases, while the relative size values increase. We argue that a modularization solution with a ration higher than 0.05 and a relative size lower than 0.01 could be considered as an acceptable solution since it could keep a good distribution of original ontology concepts among the set of generated modules. Looking to Table 11 demonstrates that the proposed partitioning tool (OAPT) to a large extent achieves this goal. The results are reported in Fig. 11.

These evaluation results show that the size of the module has strong effect on the quality of the module because the size of a module. Therefore, the OAPT tool has two different options: (i) it allows the user proposing a number of modules, or (ii) it recommends an optimal number of modules based on the proposed strategy.

- **Completeness and Consistency** During the modularization process, the proposed tool should preserve the logical consistency between every module and the original ontology [40,41], i.e., the set of classes and axioms in the module should be logically consistent with those of its original ontology [18]. To evaluate this aspect, we make use of *local correctness* and *local completeness* proposed in [18], which are implemented as simple, but effective measures in [41]. These measures are based on the difference between the original ontology entities and generated

modules’ entities. In particular, we computed the following indicators:

- *Class difference* (D_{class}); which is defined as $D_{class} = \frac{|C_{\mathcal{O}}| - |C_{\mathcal{M}}|}{|C_{\mathcal{O}}|}$, where $|C_{\mathcal{O}}|$ and $|C_{\mathcal{M}}|$ are the number of classes in the original ontology and the generated set of modules, respectively.
- *Hierarchical difference* (D_{hier}); which is defined as $D_{hier} = \frac{|isA_{\mathcal{O}}| - |isA_{\mathcal{M}}|}{|isA_{\mathcal{O}}|}$, where $|isA_{\mathcal{O}}|$ and $|isA_{\mathcal{M}}|$ are the number of subclasses (hierarchical) relations in the original ontology and the generated set of modules, respectively.
- *Non-Hierarchical difference* (D_{Nhier}); which is defined as $D_{Nhier} = \frac{|Prop_{\mathcal{O}}| - |Prop_{\mathcal{M}}|}{|Prop_{\mathcal{O}}|}$, where $|Prop_{\mathcal{O}}|$ and $|Prop_{\mathcal{M}}|$ are the number of properties (non-hierarchical) relations in the original ontology and the generated set of modules, respectively.

We computed these three metrics for each ontology in the dataset and results are reported in Fig. 12. The set of ontologies is ascending ordered on the horizontal axis based on the number of modules, such as ontologies with only one module (BP, cmt, ..., travel) are on the left side. In general, the figure shows that ontologies that generate only one module these metrics’ values are zero. The role of OAPT is to determine the number of optimal modules, and if it is one ($\mathcal{K} = 1$), OAPT keeps the original ontology without missing any information. The figure also shows that the difference in classes (D_{class}) between the original ontology and the set of generated modules has a negative value for almost of ontologies (except ontologies with one module), which means that the

Table 11 Module size evaluation

	No_module	Size_Min	Size_Max	ratio	RS
BP	1	69	69	1.000	0.000
cmt	1	30	30	1.000	0.000
Conference	1	60	60	1.000	0.000
gfo-basic	1	45	45	1.000	0.000
Ontobio	1	189	189	1.000	0.000
travel	1	35	35	1.000	0.000
AERO	2	94	106	0.887	0.001
AIDSClinic	2	14	24	0.583	0.005
asthma	2	79	185	0.427	0.004
BCO	4	31	56	0.554	0.008
CBO	4	52	141	0.369	0.007
BFO	6	3	18	0.167	0.106
ASDPTO	7	33	150	0.220	0.011
EPILONT	7	3	61	0.049	0.041
CCON	9	9	27	0.333	0.038
pl	10	14	131	0.107	0.015
TOK	10	4	59	0.068	0.034
CN	11	8	222	0.036	0.016
ROO	11	7	321	0.022	0.012
BMT	12	13	77	0.169	0.023
mouse_anatomy	12	270	447	0.604	0.001
BHN	14	101	561	0.18	0.004
gfo-bio	14	5	28	0.179	0.042
oboe-sbc	15	13	80	0.163	0.011
pato	15	201	422	0.476	0.001
SBO	15	53	111	0.477	0.006
EOL	16	39	134	0.291	0.009
PW	22	109	253	0.431	0.003
FLOPO	24	11	45	0.244	0.407
nci_anatomy	25	147	574	0.256	0.003
ADO	27	16	313	0.051	0.014
ENVO	33	132	922	0.143	0.003
CHEBI	62	422	3912	0.145	0.0004

proposed modularization approach could preserve the set of classes of the original ontology. These negative values of D_{class} demonstrate the overlapping of concepts between different modules. One more issue to be remarked here is about the *Flopo* ontology. It is the only ontology (among tested dataset) which has a positive D_{class} , which means that the proposed modularization approach misses a set of concepts from the original ontology. It misses more than 80% of the original ontology.

Figure 12 also shows that the *OAPT* tool could also to a large extent preserve hierarchical relations from the original ontology, where the values of D_{hier} are 0 or close to 0 for almost tested ontologies. However,

OAPT could not preserve the hierarchical relations from *gfo-bio*, *obo-sbc*, *Flopo* and *Chebi* ontologies. The *gfo-bio* ontology is a small ontology with 166 concepts and partitioned into 14 modules, which may be the reason behind this hierarchical relations loss (the original ontology has 179 isA relations, while the set of generated modules has only 91).

to sum up, the proposed approach could preserve the set of classes and hierarchical relations from the original ontologies, while it misses a large amount of non-hierarchical relations. It also at the same time could preserve the set of individuals and labels from the original ontology.

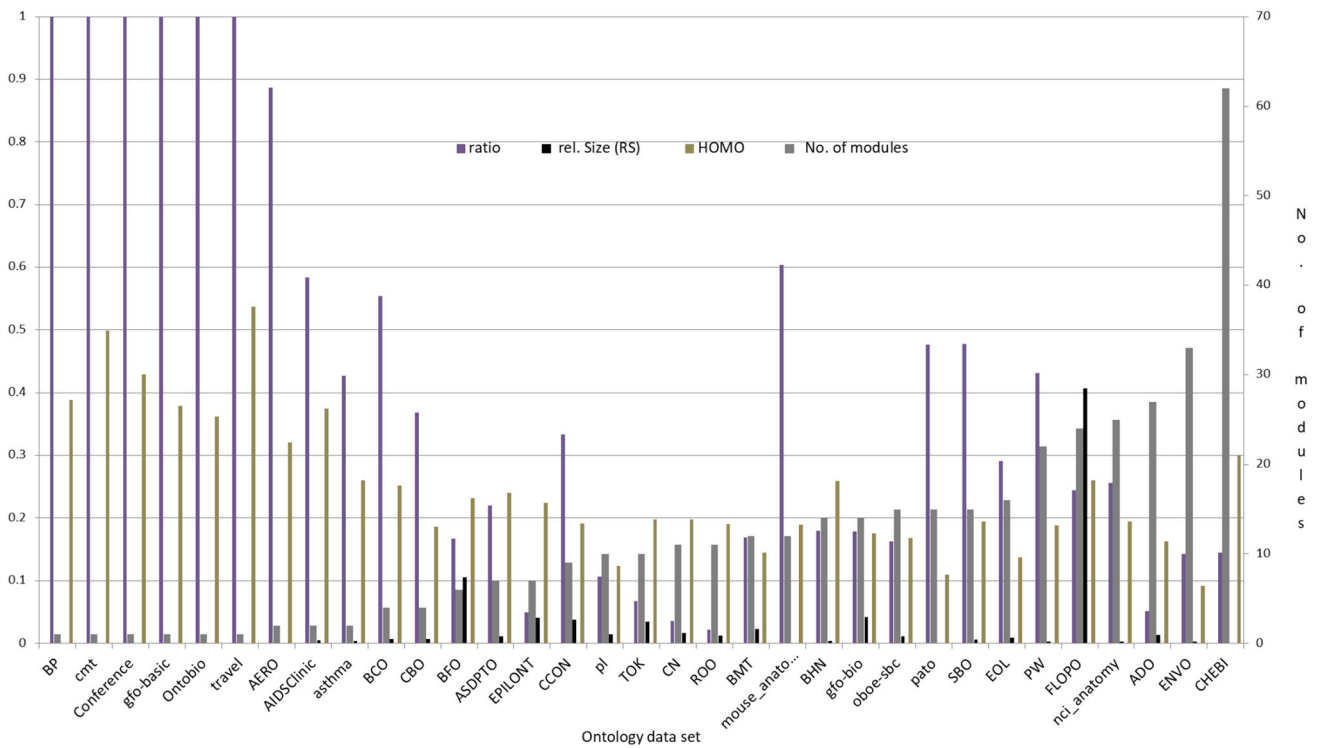


Fig. 11 Module homogeneity w.r.t. RS and ratio

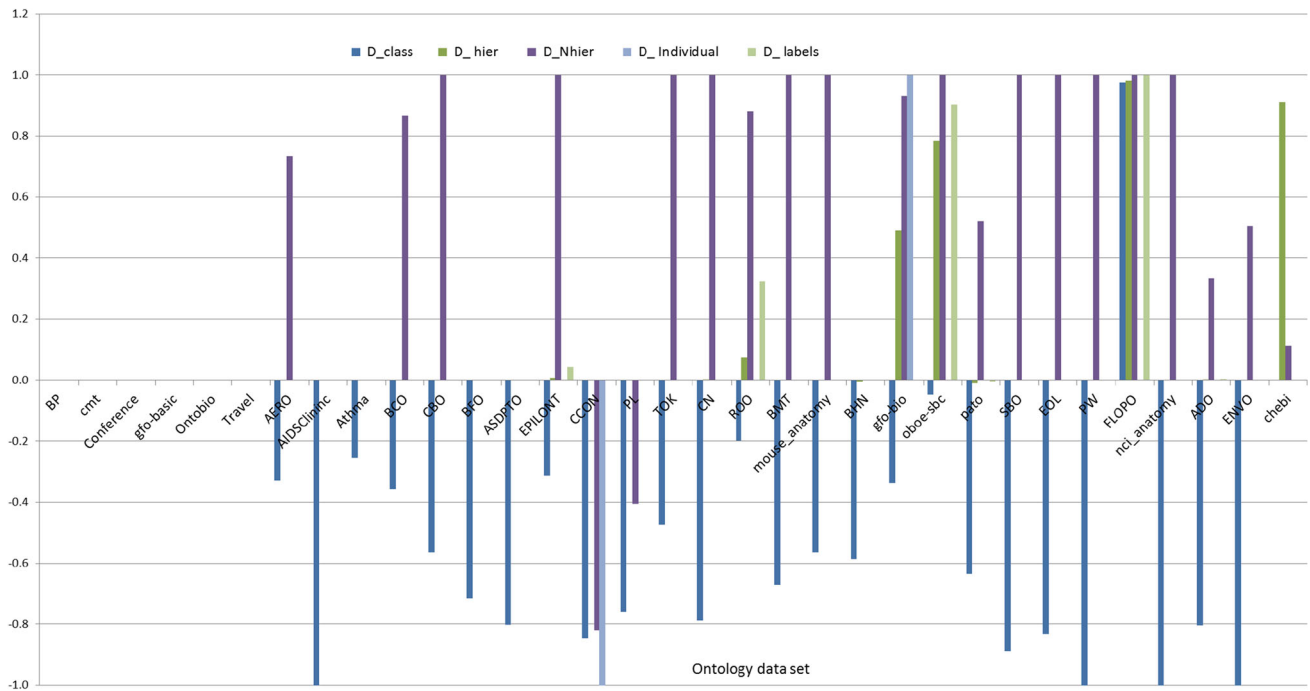


Fig. 12 OAPT completeness metrics

Tool Performance and Usability In this aspect, we aim to evaluate the functionality supporting by *OAPT* during modularization. To this end, we consider some criteria proposed in [41], such as language, modularization support, and visualization. A summary of these criteria is reported in Table 12. The table shows that our proposed tool supports the modularization of ontologies represented in different formats and languages, such as RDF, OWL, and OBO. Furthermore, *OAPT* provides different functionalities that supports the user interacts during the modularization process. For example, the user can set the number of required modules. One important feature that tool can also support is module visualization. It is very important, and a necessary feature of the tool is to allow either visualizing each module or the set of generated modules together, as shown in Figs. 4. One more important feature that *OAPT* supports w.r.t. its usability is that the tool supports download output in file format, the output file can be stored and used in other tools effectively.

Comparison

Finally, we compared our approach to two well-known approaches that are commonly used for ontology partitioning: *PATO Partitioning Tool*[51] and *PBM* [27].¹³ *PATO* is a tool for partitioning light-weight ontologies based on the graph partitioning scheme. The number of partitions to be created is automatically defined by the approach, and it depends on a set of criteria, such as the maximum number of concepts within a partition. In this comparison, we set it to 250 concepts.¹⁴ *PMB* is an agglomerative partitioning approach for ontology partitioning. It extends the *ROCK* algorithm [23]. We applied both *PATO* and *PMB* to the ontology dataset listed in Table 4 and compared their results to results produced by *OAPT*. The results are reported in Figs. 13 and 14.

Figure 13 shows the results of comparison w.r.t. the number of modules (partitions) generated by each tool. From this figure, we get several observations: i) *OAPT* (our proposed approach) could successfully partition all tested ontologies, however, both *PATO* and *PBM* failed to partition a set of these ontologies. There are different reasons behind this, for example, *PATO* fails to partition a set of ontologies (*ENVO*, *Pato*, *EOL*) for reading/parsing errors, while *PMB* can not partition *CHEBI* since it is a very big ontology. It should be mentioned that the *PBM* tool could partition a number of ontologies higher than the *PATO* tool. ii) The figure shows also that *PATO* gen-

erates a larger number of modules (partitions) compared to other two systems. For example, the *mouse_anatomy* ontology, even it has only 2743 named concepts, however, it is partitioned into 143, 100, and 12 modules using *PATO*, *PBM*, and *OAPT*, respectively. It should be noted that changing the setting of *PATO* results in completely different set of partitions. For example, set the maximum partition size to 500 results in partitioning the *mouse_anatomy* ontology into 75 partitions.

We also evaluated these two tools and compared them to *OAPT* w.r.t. the module completeness. The results are reported in Figs. 14 and 15, for the completeness of *PBM* and *PATO*, respectively. Regarding to the class difference (D_{diff}), Fig. 14 shows that the *PMB* approach produces a set of disjoint partitions, where the D_{class} values are always greater or equal to 0, while Fig. 15 illustrates that *PATO* completely preserve concepts from the original ontology. This can be explained as *PATO* generates one more module that includes non-partitioned concepts without any kind of relations among them. this module only contains a set of non-partitioned concepts. For example, *PATO* partitions the *mouse anatomy* ontology into 143 partitions. One of them only contains a set of 904 non-partitioned concepts. *PATO* fails only to preserve concepts from two *SBO* and *OBOE* ontologies, as shown in Fig. 15.

Figures 14 and 15 shows that both tools could to some extent preserve hierarchical relations from the original ontologies, where $D_{hier_{PMB}} = 0.106$ and $D_{hier_{PATO}} = 0.149$. However, both systems can not to a large extent preserve non-hierarchical relations from original ontologies, where $D_{Nhier_{PMB}} = 0.49$ and $D_{Nhier_{PATO}} = 0.65$.

To sum up, comparing to *PATO* and *PMB*, our proposed tool could manage and deal with partitioning all tested ontologies; however, these tools fail to partition some ontologies either for its large number of concepts (e.g., *chebi*) or for I/O related errors. The second point is that our proposed tool generates a reasonable number of modules for each ontology, since we first analyzed the input ontology and recommended an optimal number of modules. Regarding to the completeness metrics, *OAPT* outperforms the other two systems since it could preserve ontology information (class, hierarchical, and non-hierarchical relations) better than the two other systems.

¹³ <https://ws.nju.edu.cn/falcon-ao/>.

¹⁴ The default value for the maximum number of concepts within a partition was set in *PATO* to 300, we did such a small change to make *PATO* be able to deal with smaller ontologies.

Table 12 OAPT performance w.r.t criteria in [41]

Perspective	Criteria	OAPT	PATO
Usability	Input	Number of modules, type of partitiong	Depth of tree, isolation size
	Output	View, file	View, file
	Compatibility	Interoperable within Pajek package suite	Interoperable within Pajek package suite
	User interaction	Medium	Low
	Availability	Open repositories	Local
	Visualization	Visual interface	Visual interface
	Extensibility	Extensible for different use cases, matching and merging	No
Data performance	Quick startup	JVM	JVM
	Module size	Depending on the number of ontology concepts	Small or medium
	Module coupling	Medium	Weak
	Module cohesion	Medium	Weak
	Consistency	Consistent	Incomplete
Modularization performance	Language	OWL,RDF,OBO, ttl	OWL, RDF
	Modular options	No. of modules, type of partitioning	Depth, isolation size extraction method, partition method

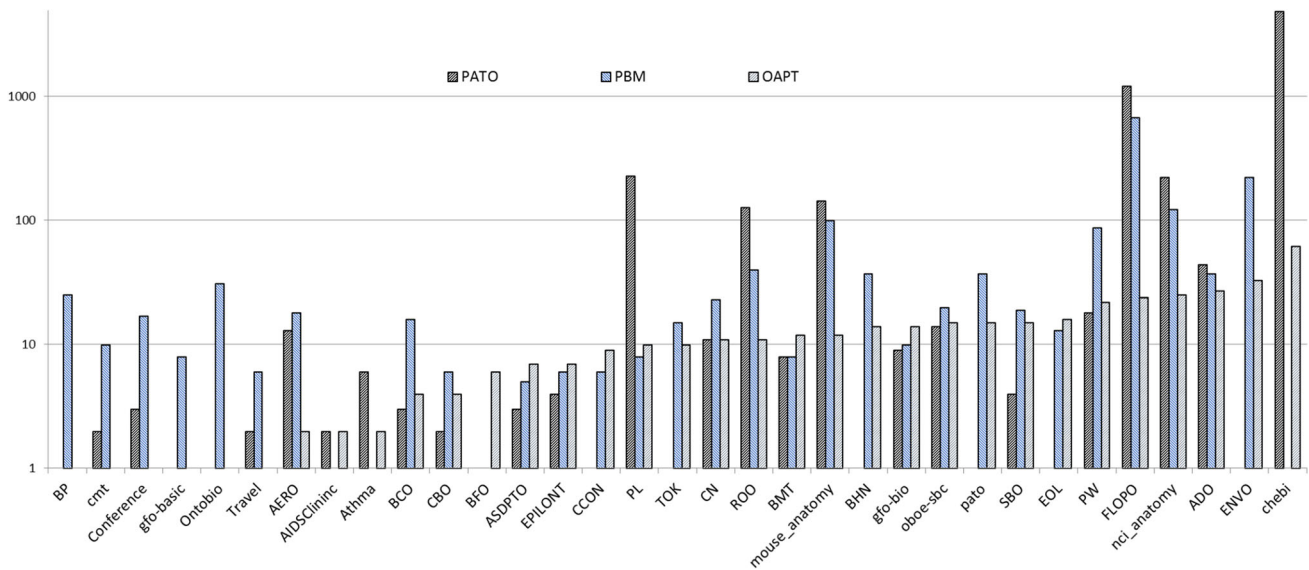


Fig. 13 Comparison between OAPT, PATO, and PBM w.r.t. the number of modules

5.2.3 User-based Evaluation

The evaluation up to now has shown that our system meets the predefined criteria and is sufficiently performant. However, the best performance numbers mean nothing, if the system does not provide satisfactory results from a user’s perspective. The latter is difficult to quantify and somewhat subjective. We thus decided on a qualitative evaluation of

this aspect involving a small number of experts. For this, we approached the main developers of *BCO*, *ENVO*, *FLOPO*, and *Study* to judge the quality of modularization. We first prepared a set of modules for each ontology using the OAPT without any guidance from their side. We asked each user to evaluate the modularization output based on their points of view. We summarize the results from this evaluation into the following two items:

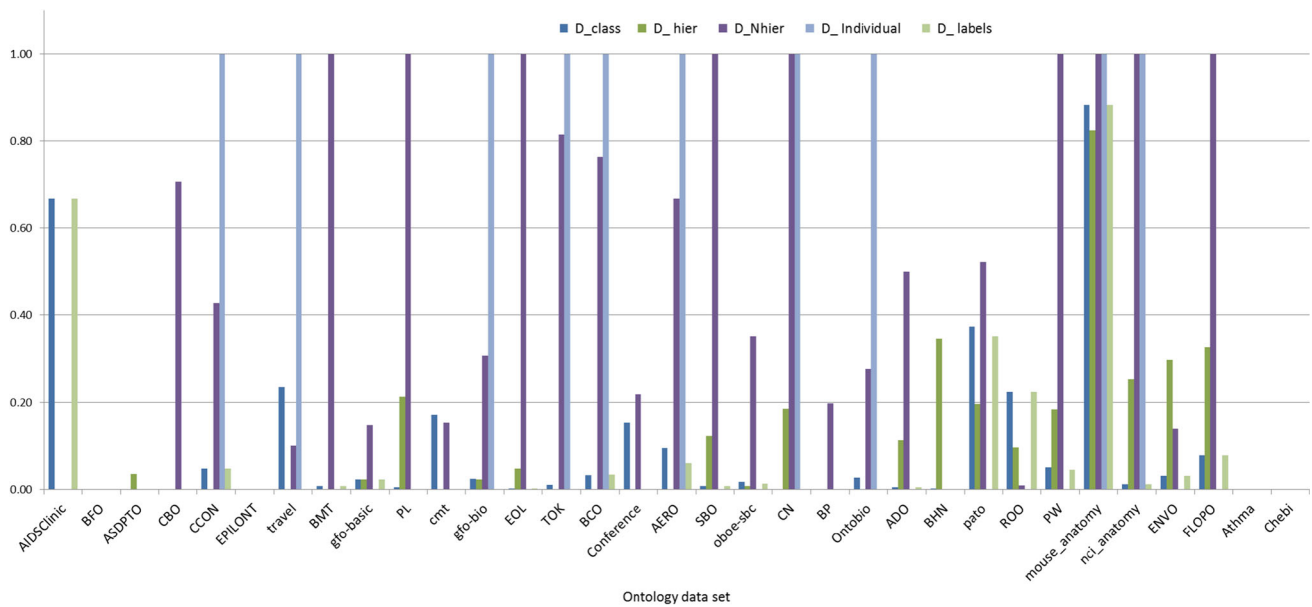


Fig. 14 PBM completeness metrics

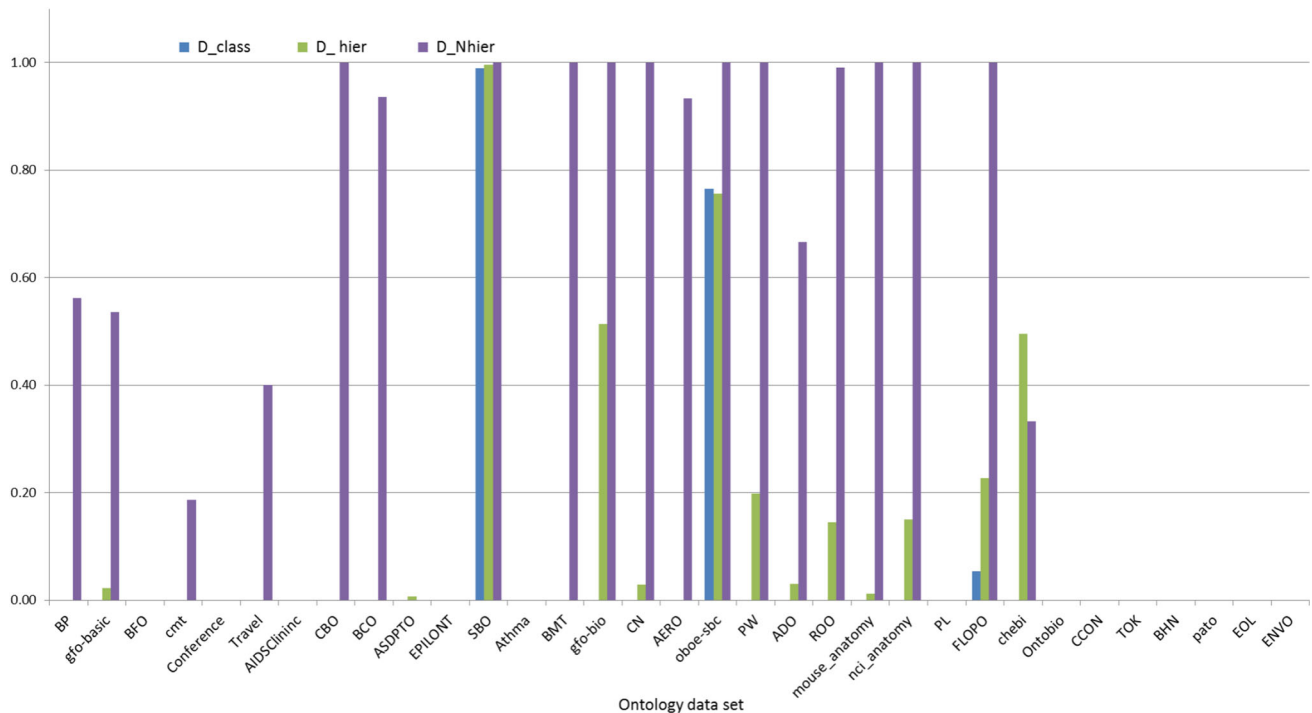


Fig. 15 PATO completeness metrics

- *Recommended number of modules* the *BCO*, *FLOPO* and *Study* ontologies' users accepted the number of modules recommended by the tool, however, the *ENVO* ontology' users see it is better if the ontology modularized into a smaller number of modules than the generated by the tool.
- *Module quality* this is the hardest task in the user-based evaluation since the user has to go through each module to

check the homogeneity of each module, as well as check the coupling between different modules. Within each module, the domain expertise investigated if a concept belonging to a partition or not. We used this information to compute the precision of each module and then compute the overall precision for the whole ontology. To compute the precision for each module, we consider the following equation: $Pre_{\mathcal{M}_i} = \frac{TP}{TP+FP}$, where TP is the

Table 13 Study ontology evaluation

Module	No. Concept	Correct	Label	Precision
1	25	21	mini or core ontology. HumanStudySubject	0.85
2	50	43	StudyActivity and Observation concepts	0.86
3	84	48	groups all sorts of outcomes	0.57
4	130	43	Activity and its properties	0.33
5	203	71	date-related concepts	0.35
total precision			0.54	

number of correct concepts belonging to the module \mathcal{M}_i and $TP + FP$ is the total number of concepts belonging to the module. For the *BCO* ontology, it contains 126 concepts and has been modularized into four modules. The domain expertise evaluation reports that the first module contains just the obsolete classes, which is not useful from the domain expertise point of view, while the second is mostly based on material entities and the upper level classes that are used to define material entity. The third module also makes sense. It is based on the processes and some of the other kinds of classes that support the definitions of processes. And finally, the last module is a useful module, based on information content entities. This one is the most internally consistent of the four. We see that this is a very good feature for the proposed tool where it is able to provide a meaningful small partition from the main ontology.

For the *ENVO* ontology, it is a bigger ontology compared to *BCO*, which makes it harder to be evaluated by the user. The ontology contains 6191 named concepts and has been modularized into 33 modules. The domain expertise provides us detailed information which enable us to compute the precision of the modularization process. *OAPT* achieves 0.79 precision when partitioning the *ENVO* ontology. The attachment files give more information about this evaluation result. Furthermore, the domain expertise could also label a set of concepts in a module to be used as an independent ontology. For the user evaluation for the *Study* ontology is reported in Table 13. The table shows that this ontology has been modularized into five modules with a total precision of 0.54. It also illustrates that the precisions of two of these modules are higher than 85%, where the first module represents a core part of the main ontology with the label “*HumanStudySubject*.” The other module represents the set of study activity concepts.

To sum up, the use-based evaluation validates the quality of the proposed modularization tool and confirms that the tool is able to generate meaningful modules which can be

independently used from the original ontology. However, it also reveals a number of rooms for improvement.

6 Conclusions

In this paper, we addressed the problem of ontology reuse. To this end, we proposed and developed a seeding-based partitioning approach that modularizes a given ontology into a set of modules. A crucial step during the modularization process is to determine how many modules should be generated. To address this question, we introduced an information-theoretic model selection algorithm that makes use of BIC to effectively select the “optimal” number of modules. To reduce the search space of the model selection step, we estimated a boundary interval. Experimental results showed that the proposed approach has the ability to modularize ontologies having different characteristics and representing different domains without the preceding knowledge of the number of modules. We also validated the performance of the approach w.r.t. a set of well-known criteria. In our going work, we need to validate the semantics of the modularization output w.r.t. the original ontology.

In future work, we want to complete the tool in order to support scientists selecting which module (or a set of modules) fulfills their requirements. Furthermore, we plan to improve the ontology analysis phase by considering more measures and criteria and to improve also the partitioning phase by taking into account other partitioning techniques. Furthermore, we plan to visualize all these processes and steps to be more user-interactive.

Acknowledgements Our sincere thanks to the domain experts (*Claus Weiland* and *Marco Schmidt*, Senckenberg Biodiversity and Climate Research Centre, Germany; *Naouel Karam*, Fraunhofer institute for Open Communication Systems, Germany; *Melanie Buss*, Center for Marine Environmental Sciences, Germany; and *Ramona Walls*, CyVerse, Bio5 Institute, University of Arizona) who participated in the evaluation and contributed a significant amount of time. Thanks are also due to the reviewers of the first version of the manuscript for their constructive comments. This work has been mostly funded by the *Deutsche Forschungsgemeinschaft (DFG)* as part of the CRC 1076 AQUADIVA. This paper is a revised and extended version of the paper presented in [2,3].

Funding Open Access funding provided by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Algergawy A, Babalou S, Kargar MJ, Davarpanah SH (2015) Seecont: A new seeding-based clustering approach for ontology matching. In: 19th International conference on advances in databases and information systems, ADBIS, pp 245–258
- Algergawy A, Babalou S, Klan F, König-Ries B (2016) OAPT: A tool for ontology analysis and partitioning. In: Proceedings of the 19th international conference on extending database technology, EDBT, pp 644–647
- Algergawy A, Babalou S, König-Ries B (2016) A new metric to evaluate ontology modularization. In: 2nd international workshop on summarizing and presenting entities and ontologies co-located with the 13th extended semantic web conference
- Algergawy A, Massmann S, Rahm E (2011) A clustering-based approach for large-scale ontology matching. In: Eder J, Bielikova M, Tjoa AM (eds) Advances in databases and information systems. ADBIS 2011. Lecture Notes in Computer Science, vol 6909. Springer, Berlin, Heidelberg, pp 415–428. https://doi.org/10.1007/978-3-642-23737-9_30
- Algergawy A, Nayak R, Saake G (2010) Element similarity measures in XML schema matching. *Inf Sci* 180(24):4975–4998
- Amato F, Santo AD, Moscato V, Persia F, Picariello A, Poccia SR (2015) Partitioning of ontologies driven by a structure-based approach. In: 2015 IEEE international conference on semantic computing, pp 320–323
- Bezerra C, de Freitas FLG, Zimmermann A, Euzenat J (2008) Mod-onto: A tool for modularizing ontologies. In: Proceedings of the 3rd workshop on ontologies and their applications
- Bodenreider O (2008) Biomedical ontologies in action: Role in knowledge management, data integration and decision support. *Yearb Med Inform* 2008(suppl 1):67–79
- Burton-Jones A, Storey VC, Sugumaran V, Ahluwalia P (2005) A semiotic metrics suite for assessing the quality of ontologies. *Data Knowl Eng* 55(1):84–102
- d'Aquin M, Baldassare C, Gridinoc L, Angeletou S, Sabou M, Motta E (2007) Characterizing knowledge on the semantic web with watson. In: 5th international workshop on evaluation of ontologies and ontology-based tools, pp 1–10
- d'Aquin M, Schlicht A, Stuckenschmidt H, Sabou M (2007) Ontology modularization for knowledge selection: Experiments and evaluations. In: 18th international conference on database and expert systems applications, DEXA, pp 874–883
- Doran P, Tamma VAM, Iannone L, Ontology module extraction for ontology reuse: an ontology engineering perspective. In: Proceedings of the sixteenth ACM conference on information and knowledge management, CIKM, pp 61–70
- Duque-Ramos A et al (2013) Evaluation of the oquare framework for ontology quality. *Expert Syst. Appl.* 40(7):2696–2703
- Fernandez M, Cantador I, Castells P (2006) Core: A tool for collaborative ontology reuse and evaluation. In: 4th international workshop on evaluation of ontologies for the web (EON 2006)
- Freeman LC (1997) A set of measures of centrality based on betweenness. *Sociometry*
- Gangemi A, Catenacci C, Ciaramita M, Lehmann J (2006) Modelling ontology evaluation and validation. In: 3rd european semantic web conference, ESWC, pp 140–154
- Grau BC, Horrocks I, Kazakov Y, Sattler U (2007) Just the right amount: extracting modules from ontologies. In: Proceedings of the 16th international conference on world wide web, WWW, pp 717–726
- Grau BC, Horrocks I, Kazakov Y, Sattler U (2007) A logical framework for modularity of ontologies. In: 20th international joint conference on artificial intelligence, IJCAI, pp 298–303
- Grau BC, Horrocks I, Kazakov Y, Sattler U (2008) Modular reuse of ontologies: Theory and practice. *J Artif Intel Res (JAIR)* 31:273–318
- Grau BC, Parsia B, Sirin E, Kalyanpur A (2005) Automatic partitioning of OWL ontologies using *E*-connections. In: proceedings of the 2005 international workshop on description logics (DL2005)
- Grau BC, Parsia B, Sirin E, Kalyanpur A (2006) Modularity and web ontologies. In: Proceedings, tenth international conference on principles of knowledge representation and reasoning, lake district of the United Kingdom, June 2–5, 2006, pp 198–209
- Guarino N, Oberle D, Staab S (2009) What is an ontology? In: Staab S, Studer R (eds) Handbook on Ontologies. International handbooks on information systems. Springer, Berlin, Heidelberg, pp 1–17. https://doi.org/10.1007/978-3-540-92673-3_0
- Guha S, Rastogi R, Shim K (2000) ROCK: A robust clustering algorithm for categorical attributes. *Inf. Syst.* 25(5):345–366
- Heino N, Pan JZ (2012) RDFS reasoning on massively parallel hardware. In: The semantic web - ISWC 2012 - 11th international semantic web conference, pp 133–148
- Hendler J (2001) Agents and the semantic web. *IEEE Intel Syst J* 16:30–37
- Horrocks I (2008) Ontologies and the semantic web. *Commun. ACM* 51(12):58–67
- Hu W, Qu Y, Cheng G (2008) Matching large ontologies: A divide-and-conquer approach. *DKE* 67:140–160
- Ishioka T (2000) Extended k-means with an efficient estimation of the number of clusters. In: Second international conference on intelligent data engineering and automated learning - IDEAL, pp 17–22
- Kalyanpur A, Parsia B, Sirin E, Grau BC, Hendler JA (2006) Swoop: A web ontology editing browser. *J Web Semant* 4(2):144–153
- Khan ZC (2016) Evaluation metrics in ontology modules. In: Proceedings of the 29th international workshop on description logics, cape town, south africa, April 22–25, 2016
- Khan ZC, Keet CM (2015) Toward a framework for ontology modularity. In: Proceedings of the 2015 annual research conference on south african institute of computer scientists and information technologists, SAICSIT '15, pp 24:1–24:10
- Konev B, Lutz C, Walther D, Wolter F (2013) Model-theoretic inseparability and modularity of description logic ontologies. *Artif Intell* 203:66–103
- Kothari R, Pitts D (1999) On finding the number of clusters. *Pattern Recognit Lett* 20(4):405–416
- Lantow B (2016) Ontometrics: Application of on-line ontology metric calculation. In: Joint proceedings of the BIR 2016 workshops and doctoral consortium co-located with 15th international conference on perspectives in business informatics research (BIR 2016), Prague, Czech Republic, September 14 - 16, 2016

35. Li N, Motta E, d'Aquin M (2010) Ontology summarization: an analysis and an evaluation. In: Proceedings of the international workshop on evaluation of semantic technologies (IWEST 2010), Shanghai, China, November 8, 2010
36. Ma Y, Lu K, Zhang Y, Jin B (2013) Measuring ontology information by rules based transformation. *Knowl-Based Syst* 50:234–245
37. Marwah K, Katzin D, Zollanvari A, Ramoni NNM, Alterovitz G (2012) Context-specific ontology integration: A bayesian approach. In: MIA summits on translational science proceedings, pp 79–86
38. Milligan GW, Cooper MC (1985) An examination of procedures for determining the number of clusters in a data set. *Psychometrica* 50:159–179
39. Myung IJ (2003) Tutorial on maximum likelihood estimation. *J Math Psychol* 47:90–100
40. Oh S, Ahn J (2009) Ontology module metrics. In: 2009 IEEE International conference on e-Business engineering, ICEBE 2009, Macau, China, 21–23 October 2009, pp 11–18
41. Oh S, Yeom HY (2012) A comprehensive framework for the evaluation of ontology modularization. *Expert Syst Appl* 39(10):8547–8556
42. Parkinson D, Liddle AR (2013) Bayesian model averaging in astrophysics: a review. *Stat Anal Data Mining* 6(1):3–14
43. Pathak J, Johnson TM, Chute CG (2009) Survey of modular ontology techniques and their applications in the biomedical domain. *Integr Comput-Aided Eng* 16(3):225–242
44. Pham DT, Dimov SS, Nguyen CD (2005) Selection of K in K-means clustering. *Mech Eng Sci* 219(3):103–118
45. Priya S, Guo Y, Spear M, Heflin J (2014) Partitioning OWL knowledge bases for parallel reasoning. In: 2014 IEEE international conference on semantic computing, pp 108–115
46. Quesada-Martínez M, Duque-Ramos A, Fernández-Breis JT (2015) Analysis of the evolution of ontologies using oquare: application to EDAM. In: Proceedings of the international conference on biomedical ontology, ICBO
47. Quesada-Martínez M, Fernández-Breis JT, Stevens R, Aussenac-Gilles N (2014) Ontoenrich: A platform for the lexical analysis of ontologies. In: Knowledge engineering and knowledge management — EKAW 2014 satellite events, VISUAL, EKM1, and ARCOE-Logic, pp 172–176
48. Romero AA, Kaminski M, Grau BC, Horrocks I (2015) Ontology module extraction via datalog reasoning. In: 29th AAAI conference on artificial intelligence, pp 1410–1416
49. Romero AA, Kaminski M, Grau BC, Horrocks I (2016) Module extraction in expressive ontology languages via datalog reasoning. *J. Artif. Intell. Res. (JAIR)* 55:499–564
50. Schlicht A, Stuckenschmidt H (2006) Towards structural criteria for ontology modularization. In: 1st International workshop on modular ontologies, WoMO'06, co-located with the international semantic web conference, ISWC'06
51. Schlicht A, Stuckenschmidt H (2008) A flexible partitioning tool for large ontologies. In: International conference on web intelligence, WI, pp 482–488
52. Schwarz G (1978) Estimating the dimension of a model. *Ann Stat* 6(2):461–454
53. Seidenberg J, Rector AL (2006) Web ontology segmentation: analysis, classification and use. In: 15th international conference on world wide web, pp 13–22
54. Sicilia M, Rodríguez D, Barriocanal EG, Alonso SS (2012) Empirical findings on ontology metrics. *Expert Syst. Appl.* 39(8):6706–6711
55. Smith A (2004) Model selection: Bayesian information criterion. *Encycl Stat Sci*, 8
56. Steinberg MD, Schindler S, Keil JM (2016) Use cases and suitability metrics for unit ontologies. In: 13th OWL: experiences and directions workshop and 5th OWL reasoner evaluation workshop
57. Stoilos G, Stamou GB, Kollias SD (2005) A string metric for ontology alignment. In: The semantic web - ISWC 2005, 4th international semantic web conference, ISWC 2005, Galway, Ireland, November 6–10, 2005, Proceedings, pp 624–637
58. Stuckenschmidt H, Klein MCA (2004) Structure-based partitioning of large concept hierarchies. In: Third international semantic web conference, ISWC 2004, pp 289–303
59. Studer R, Benjamins VR, Fensel D (1998) Knowledge engineering: Principles and methods. *Data Knowl. Eng.* 25(1–2):161–197
60. Tartir S, Arpinar IB (2007) Ontology evaluation and ranking using OntoQA. In: First IEEE international conference on semantic computing (ICSC), pp 185–192
61. Troullinou G, Kondylakis H, Daskalaki E, Plexousakis D (2017) Ontology understanding without tears: The summarization approach. *Semantic Web* 8(6):797–815
62. Vescovo CD, Gessler D, Klinov P, Parsia B, Sattler U, Schneider T, Winget A (2011) Decomposition and modular structure of bioportal ontologies. In: 10th international semantic web conference ISWC proceedings, Part I, pp 130–145
63. Vescovo CD, Parsia B, Sattler U, Schneider T (2011) The modular structure of an ontology: Atomic decomposition. In: IJCAI, pp 2232–2237. IJCAI/AAAI
64. Wu Z, Palmer M (1994) Verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on association for computational linguistics, pp 133–138
65. Yan M (2005) Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion. PhD thesis, Virginia Polytechnic Institute and State University
66. Zhang H, Li Y, Tan HKB (2010) Measuring design complexity of semantic web ontologies. *J Syst Softw* 83(5):803–814

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.