Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 19(7): 7388–7409.
DOI: 10.3934/mbe.2022349
Received: 05 December 2021
Revised: 02 April 2022
Accepted: 08 April 2022
Published: 19 May 2022

Research article

Effective method for detecting error causes from incoherent biological ontologies

Yu Zhang[1,2,3], Haitao Wu[1,2], Jinfeng Gao[1,2], Yongtao Zhang[4], Ruxian Yao[1,2,*] and Yuxiang Zhu[1,2]

[1] College of Information Engineering, Huanghuai University, Zhumadian 463000, China
[2] Henan Key Laboratory of Smart Lighting, Zhumadian 463000, China
[3] Henan Joint International Research Laboratory of Behavior Optimization Control for Smart Robots, Zhumadian 463000, China
[4] Department of Information and Electronic Engineering, Shangqiu Institute of Technology, Shangqiu 476000, China

* **Correspondence:** Email: yaoruxian@126.com.

Abstract: Computing the minimal axiom sets (MinAs) for an unsatisfiable class is an important task in incoherent ontology debugging. Ddebugging ontologies based on patterns (DOBP) is a pattern-based debugging method that uses a set of heuristic strategies based on four patterns. Each pattern is represented as a directed graph and the depth-first search strategy is used to find the axiom paths relevant to the MinAs of the unsatisfiable class. However, DOBP is inefficient when a debugging large incoherent ontology with a lot of unsatisfiable classes. To solve the problem, we first extract a module responsible for the erroneous classes and then compute the MinAs based on the extracted module. The basic idea of module extraction is that rather than computing MinAs based on the original ontology $\mathcal{O}$, they are computed based on a module $\mathcal{M}$ extracted from $\mathcal{O}$. $\mathcal{M}$ provides a smaller search space than $\mathcal{O}$ because $\mathcal{M}$ is considerably smaller than $\mathcal{O}$. The experimental results on biological ontologies show that the module extracted using the Module-DOBP method is smaller than the original ontology. Lastly, our proposed approach optimized with the module extraction algorithm is more efficient than the DOBP method both for large-scale ontologies and numerous unsatisfiable classes.

Keywords: minimal axioms sets; unsatisfiable class; incoherent ontology; DOBP; module-DOBP

1. Introduction

Description logics (DLs) [1] are a family of logic-based knowledge representation formalisms that can be used to develop ontologies using the web ontology language (OWL) [2]. DL ontology typ-

ically consists of TBox axioms and ABox axioms. TBox axioms represent relationships between classes or between properties. For example, the axiom "mitochondrion $\sqsubseteq$ cytoplasm" states that the class mitochondrion is the subclass of cytoplasm, while the axiom "hasChild $\sqsubseteq$ hasSibling" states that the property hasChild is a sub-property of hasSibling. ABox axioms represent relationships between classes and individuals, and between properties and individuals. For example, LocatedIn(mitochondria,cytoplasm) states that mitochondria are located in the cytoplasm. This study focuses on the TBox part of ontologies. Given that the TBox reasoning is not influenced by ABox reasoning [3], we assume that an ontology consists of only a TBox in the rest of the study. In the opening environment, errors often occur in a biological ontology when the same ontology is simultaneously edited by more than one participators, and the majority of them are unaware of the existences of one another [4]. Errors mean that the definitions of some of the classes yield logical conflicts. We call the classes as unsatisfiable classes and the ontology as incoherent ontology.

The following is a example of an incoherent gene ontology $\mathcal{O}_1$.

α_1: mitochondrion $\sqsubseteq$ organelle (mitochondrion is an organelle)

α_2: mitochondrion $\sqsubseteq$ cytoplasm (mitochondrion is a part of cytoplasm)

α_3: cytoplasm $\sqsubseteq$ cell (cytoplasm is a part of a cell)

α_4: mitochondrion $\sqsubseteq \neg$ cell (mitochondrion is not a part of a cell)

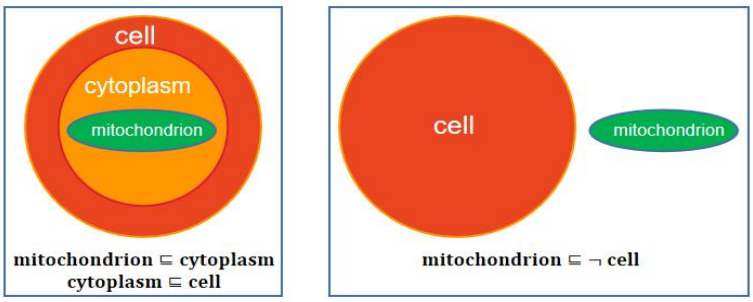


Figure 1. Illustration show how thedefinition of mitochondrion yields a logical conflict.

Biological proteins play an important role in most cellular processes, such as gene regulation, recombination, repair, replication, and DNA modification [5]. Research on biological proteins has effectively contributed to interventions and cancer therapies [6, 7]. Given the increasing number of biological ontologies developed on the semantic web, manually finding the cause of errors has becomes a significantly difficult task. Ontology debugging services can find the reason why a certain error occurs by computing the minimal axiom sets (MinAs) related to the error. This helps developers and users of biological ontology to understand the reason why an error follows from the ontology. Therefore, debugging an incoherent ontology is performed to determine the reasons why the classes in the ontology are unsatisfiable. Given the previously mentioned ontology $\mathcal{O}_1$, we can deduce that mitochondrion $\sqsubseteq$ cell (i.e., mitochondrion is a part of a cell) according to α_2 and α_3. However, mitochondrion is known to not be a part of a cell according to α_4. At this point, we can conclude that the definition of mitochondrion yields a logical conflict, as shown in Figure 1. Therefore, mitochondrion is an unsatisfiable class and $\mathcal{O}_1$ is an incoherent ontology. The aim of ontology debugging is to find which axioms cause the unsatisfiability of mitochondrion. The result is MinAs($\mathcal{O}_1$, mitochondrion) = $\{\alpha_2, \alpha_3, \alpha_4\}$.

MinAs is a common debugging technique, also known as justifications and minimal unsatisfiability-

preserving sub-TBoxes (MUPS). Justification is a minimal set of axioms of an incoherent ontology that can explain an unsatisfiable class. The MUPS refers to the smallest subsets of axioms of an incoherent ontology preserving unsatisfiability of an unsatisfiable class [8]. Two types of methods for computing MinAs (or Justifications and MUPS) are the greasoner-independent-based glass-box methods and the reasoner-dependent black-box methods. The former modifies the internal tableau-based algorithm of the reasoner, thereby making these methods dependent but with a limited portability [9]. By contrast, the latter adopts an "expand-contract" strategy to check which subset of an ontology is a MinAs without modifying the reasoner.

A significant number of studies have been devoted to the graph-based debugging methods. Reference [10] presents a consequence-based reasoning algorithm based on the notion of a *decomposition*, i.e., a graph-like structure that can capture the essential features. In Reference [11], the authors construct an explanation dependency graph from the classification result of a reasoner and then compute all justifications thereafter based on the graph. In addition, Reference [12] presents a graph-based method for debugging and revising incoherent DL-Lite ontologies. The authors first encoded DL-Lite ontology into a directed graph and then calculated the minimal incoherence-preserving path-pairs based on the directed graph.

In recent years, there have been significant research efforts devoted to studying ontology modularization. Modularization is particularly beneficial for ontology reuse [13, 14]. It is also used for the subsumption reasoning tasks and incremental classification [15]. A selection function algorithm is given in Reference [16] to compute all justifications. It was further optimized in [17] using the module extraction. The authors of Reference [18] focused on the $\mathcal{EL}$+ ontologies by extracting the reachability-based module. A goal-directed extraction method is given in Reference [19]; it was developed by backward traversing a set of axioms responsible for the given entailment. A decomposition-based extraction algorithm is proposed in Reference [20]. With their algorithm, the computation of MIS (Minimal incoherent sub-ontology) can be separately performed in each extracted modules. Moreover, a new strategy based on a local search technique is proposed in Reference [21]; it allows the user to compute the approximating core before extracting the precise minimally unsatisfiable subformulas.

The authors of Reference [9] constructed unsatisfiable dependent paths to avoid unnecessary nondeterministic expansion. The advantage is that all irrelevant axioms are not in the dependent paths and these axioms are not selected to participate in the computation of MUPS. However, such a method is only fit for an $\mathcal{ALC}$ ontology. Thereafter, the method was extended to the work in Reference [22]. The authors first extracted a clash module from the ontology and then identified the root unsatisfiable concepts from the clash module. Thereafter, MUPS of each root unsatisfiable concept can be calculated on the basis of the clash module. However, the real-world ontologies are often dynamic and modified frequently. Thus, logical errors inevitably occur in the dynamic environments. To solve this problem, a heuristic strategy was developed to reuse the previous debugging results for subsequent debugging to avoid recomputing the MUPS [23].

Seven criteria are proposed in Reference [24] to systematically compare the existing ontology debugging methods; additionally, a set of beneficial suggestions are provided for users to choose an appropriate debugging approach according to their needs. Thereafter, the research was expanded in Reference [25]. The authors evaluated the existing ontology debugging systems based on numerous ontologies with various sizes and expressivities, providing several suggestions thereafter for users or developers to choose an effective debugging algorithm or design an appropriate debugging system. In

Reference [26], Ye et al. first created a recursive expansion procedure and then explored the critical axioms one by one. Furthermore, the authors proposed an incremental reasoning procedure to substitute for a series of standard reasoning tests with respect to satisfiability. The authors of Reference [27] computed all MUPSes based on the duality between the MUPS and minimal correctness-preserving subset (MCPS) by applying parallel strategies. The MCPS represents the minimal diagnosis of a concept required to debug unsatisfiable concepts in the ontology debugging domain.

Other researchers have focused on using the fine-grained approach to resolve unsatisfiable classes. The authors of Reference [3] revised the tableaux algorithm to rewrite logical erroneous axioms and determine the parts of the axioms responsible for errors. A fine-grained method was also developed by modifying one axiom to zero or more axioms [28].

2. Preliminaries

This section elaborates the syntax and semantics of OWL DL ontologies and presents the formal definitions of debugging incoherent ontologies.

2.1. Syntax and semantics of ontologies

DL provides a set of so-called constructors, which are used to form complex classes and properties. Table 1 lists the logic constructors and their corresponding syntax and semantics.

Table 1. Syntax and semantics of OWL DL ontologies.

Constructors	**Syntax**	**Semantics**
top class	$\top$	$\Delta^{\mathcal{I}}$
bottom class	$\bot$	$\varnothing$
class name	A	$A^{\mathcal{I}}$
negation	$\neg B$	$\Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in r^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
at-least restriction	$\geq nS.C$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y : (x,y) \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \geq n\}$
at-most restriction	$\leq nS.C$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y : (x,y) \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \leq n\}$
property name	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse property	R^{-}	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}$
transitivity	$\mathsf{Trans}(R)$	$(x,y),(y,z) \in R^{\mathcal{I}} \rightarrow (x,z) \in R^{\mathcal{I}}$
property inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
class inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
class equivalence	$\mathrm{C} \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$

In Table 1, A and B are atom classes that correspond in first-order logic to unary predicates; C and D are (possibly complex) class expressions that can be recursively constructed based on the atom classes A and B using Boolean operators ($\sqcap, \sqcup, \neg$), value restrictions ($\exists R.C, \forall R.C$) and number restrictions ($\geq nS.C, \leq nS.C$) for n a non-negative integer.

Ontology comprises finite axioms with the form of the property transitivity $\mathsf{Trans}(r)$, the class inclusion $C \sqsubseteq D$, and the property inclusion $R \sqsubseteq S$. The equivalent axiom $C \equiv D$ is transformed into $C \sqsubseteq D$ and $D \sqsubseteq C$.

Logic constructors in an ontology determine the expressivity of the ontology. Expressivity $\mathcal{ALC}$ consists of the constructors $\neg A$ (negation), $C \sqcap D$ (conjunction), $C \sqcup D$ (disjunction), $\exists r.C$ (existential restriction), and $\forall r.c$ (value restriction). Expressivity $\mathcal{S}$ consists of $\mathcal{ALC}$ and transitivity. Other expressivities are the combinations of $\mathcal{ALC}$, $\mathcal{S}$, and the following expressivity symbols: $\mathcal{H}$ (role hierarchies), $\mathcal{I}$ (inverse roles), $\mathcal{O}$ (nominals), $\mathcal{F}$ (functional roles) and $\mathcal{D}$ (data type). The DL language considered in our spans $\mathcal{ALCH}$ through to $\mathcal{SHOIF}(\mathcal{D})$.

Baader et al. defined an interpretation $\mathcal{I}$ in [1] to represent the semantics of OWL DL ontologies. $\mathcal{I}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every class A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every property R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For example, we say that C is subsumed by D, and write $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.

2.2. Debugging incoherent ontologies

The unsatisfiability of a class C indicates that the definition of C in the ontology $\mathcal{O}$ is incorrect. By asking a reasoner to check whether or not C is unsatisfiable with regards to $\mathcal{O}$ (i.e., it can be expressed as $\mathcal{O} \models C \sqsubseteq \bot$), we can determine whether C is unsatisfiable [29].

Definition 1. (Unsatisfiable class) [30] A class C in an ontology $\mathcal{O}$ is unsatisfiable if and only if for each interpretation $\mathcal{I}$ of $\mathcal{O}$, $C^{\mathcal{I}} = \varnothing$.

Definition 2. (Incoherent ontology) [30] An ontology $\mathcal{O}$ is incoherent if and only if there exists at least one unsatisfiable class in $\mathcal{O}$.

Definition 3. (Inconsistent ontology) [30] An ontology $\mathcal{O}$ is inconsistent if and only if it has no interpretation.

The incoherence can be considered as a kind of the inconsistency in the TBox, i.e., the terminology part of an ontology. An incoherent ontology has an incoherent TBox. However, an ontology being inconsistent does not necessarily imply that it is coherent [30].

Figure 2 shows four examples of incoherence and inconsistency. The detailed explanations are as follows.

Figure 2(A) shows a coherent but inconsistent ontology because the two disjoint classes C_1 and C_2 share an individual a.

Figure 2(B) shows an incoherent but consistent ontology because the two disjoint classes C_1 and C_2 share a subclass C_3.

Figure 2(C) shows an incoherent and inconsistent ontology because the two disjoint classes C_1 and C_2 share a subclass C_3, which has an individual a.

Figure 2(D) shows a coherent but inconsistent TBox because the two disjoint classes C_1 and C_2 share a subclass that is a nominal $\{a\}$.

Given that the TBox reasoning is not influenced by ABox reasoning [3], we assume that an ontology consists of only a TBox in the rest of the study.

The unsatisfiability of a class can be determined using a DL reasoner, such as Pellet [31], HermiT [32], or FaCT++ [33].

Consider the following inclusion axioms that hold in our example ontology $\mathcal{O}_1$:

$\alpha_1 : B_1 \sqsubseteq A_1 \sqcap \neg A_1$ (B_1 is a part of the conjunction of A_1 and the negation of A_1)

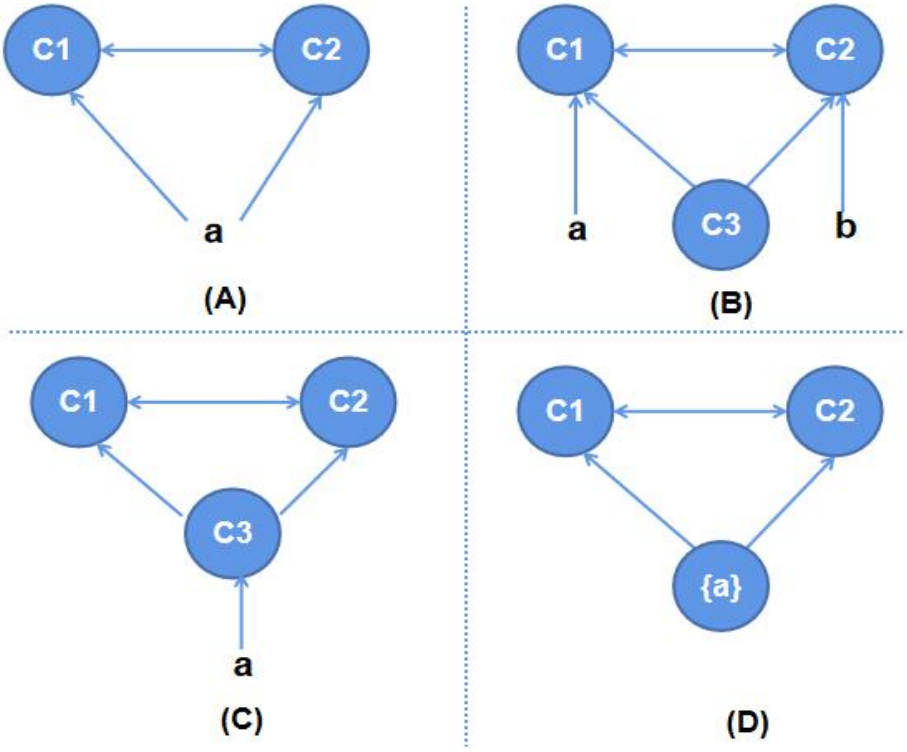


Figure 2. Examples of incoherence-and-inconsistency in [30].

$\alpha_2 : B_2 \sqsubseteq \neg A_1$ (B_2 is a part of the negation of A_1)

$\alpha_3 : C \sqsubseteq B_1 \sqcap B_2 \sqcap A_1$ (C is a part of the conjunction of B_1, B_2 and A_1)

$\alpha_4 : D \sqcap C \sqsubseteq E \sqcup F$ (the conjunction of D and C is a part of the disjunction of E and F)

$\alpha_5 : E \sqsubseteq A_1 \sqcap A_2$ (E is a part of the conjunction of A_1 and A_2)

$\alpha_6 : D \sqsubseteq C \sqcup E$ (D is a part of the disjunction of C and E)

Definition 4. (MinA) Let $\mathcal{O}$ be an incoherent ontology, and C an unsatisfiable class [34]. The set $\Sigma \subseteq \mathcal{O}$ is a minimal axiom set (MinA) for $\mathcal{O} \models C \sqsubseteq \bot$ if, and only if, for every $\Sigma' \subset \Sigma$, $\Sigma' \not\models C \sqsubseteq \bot$.

The three axioms labelled with α_1 to α_3 entail $B_1 \sqsubseteq \bot$ and $C \sqsubseteq \bot$. We can find one MinA for $B_1 \sqsubseteq \bot$: $\{\alpha_1\}$, and two MinAs for $C \sqsubseteq \bot$: $\{\alpha_1, \alpha_3\}$ and $\{\alpha_2, \alpha_3\}$. That is, MinAs($\mathcal{O}_1, B_1$)=$\{\{\alpha_1\}\}$ and MinAs($\mathcal{O}_1, C$)=$\{\{\alpha_1, \alpha_3\}, \{\alpha_2, \alpha_3\}\}$.

Algorithm 1: Calculating MinA.

Input: $\mathcal{O}$, unsatisfiable class C
Output: MinA($\mathcal{O}, C$)

```
1  Σ := O
2  for each axiom α ∈ O do
3     if Σ\α ⊨ C ⊑ ⊥ then
4        Σ := Σ\α
5  return Σ
```

Algorithm 1 was introduced in [34] to calculate one MinA of the unsatisfiable C. First, we make a copy of an ontology O as Σ (line 1). Second, for each axiom $\alpha \in O$, if the given unsatisfiable class C is unsatisfiable with regards to $\Sigma \setminus \alpha$, then we can conclude that α is not responsible for the unsatisfiability of C. Thus, we remove α from Σ (lines 2–4). After all axioms in O are tested, we can obtain an MinA of C. That is, MinA(O, C)=Σ.

However, Algorithm 1 can only calculate one MinA. Accordingly, we need to use the classic hitting set tree (HST) method if we want to calculate all MinAs. This process is shown in Figure 3.

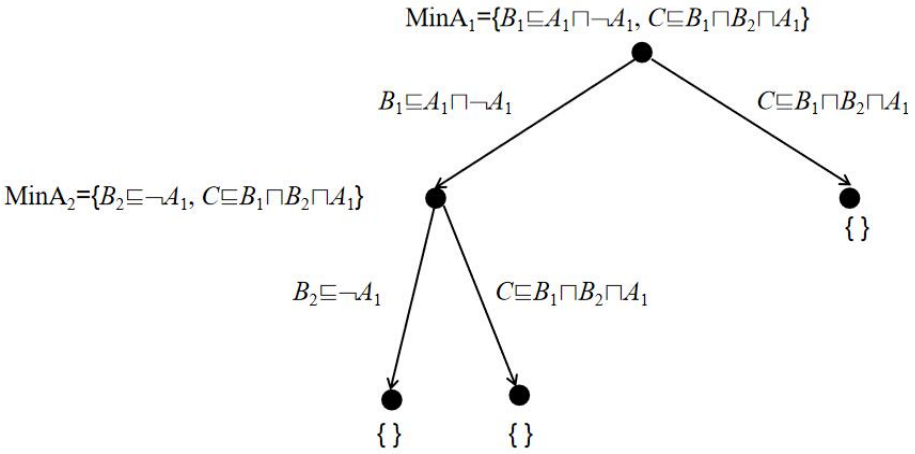


Figure 3. Process for calculating all MinAs(O_1, C) using HST method.

In Figure 3, the first MinA is computed by Algorithm 1. That is, MinA$_1$=$\{B_1 \sqsubseteq A_1 \sqcap \neg A_1, C \sqsubseteq B_1 \sqcap B_2 \sqcap A_1\}$. Taking the root node, which is labelled with MinA$_1$, the HST was extended to the left hand side by removing $B_1 \sqsubseteq A_1 \sqcap \neg A_1$ from O and computing the second MinA for $O \setminus \{B_1 \sqsubseteq A_1 \sqcap \neg A_1\}$. In this case, MinA$_2$=$\{B_2 \sqsubseteq \neg A_1, C \sqsubseteq B_1 \sqcap B_2 \sqcap A_1\}$ was found. The left hand side successor node of the root node was therefore labelled with MinA$_2$ and its connecting edge labelled with $B_1 \sqsubseteq A_1 \sqcap \neg A_1$. Similarly, the HST was extended to the left hand side by removing $B_1 \sqsubseteq A_1 \sqcap \neg A_1$ and $B_2 \sqsubseteq \neg A_1$ from O and computing the third MinA for $O \setminus \{B_1 \sqsubseteq A_1 \sqcap \neg A_1, B_2 \sqsubseteq \neg A_1\}$. However, MinA$_3 = \varnothing$.

The algorithm repeats this process by again removing an axiom, adding a node and executing Algorithm 1 to compute a new MinA. When no more successor nodes can be generated, the HST is complete. At this point, all MinAs occur as labels of nodes in the tree. That is, MinAs(O_1, C)=$\{$ MinA$_1$, MinA$_2$ $\}$ =$\{\{B_2 \sqsubseteq \neg A_1, C \sqsubseteq B_1 \sqcap B_2 \sqcap A_1\}, \{B_2 \sqsubseteq \neg A_1, C \sqsubseteq B_1 \sqcap B_2 \sqcap A_1\}\}$.

3. Proposed modularization-based DOBP algorithm

3.1. Modularization algorithm

In general, a module $\mathcal{M}$ of an ontology O comprises meaningful fragments of O that have some desirable properties [17]. For example, an incoherent module $\mathcal{M}$ for an incoherent ontology O and an unsatisfiable class C is a subset of O that is guaranteed to preserve the unsatisfiability of O.

Let a signature $\mathcal{S}$ of a DL be the union of a set of atomic classes $(A, B, ...)$ representing sets of elements and a set of properties $(R, S, ...)$ representing binary relations between elements. And let $Rol(\mathcal{S})$ be the set of properties $(R, S, ...)$ for a signature $\mathcal{S}$. For the computation of MinAs, we modify

the definition of module given in [14] as follows.

Definition 5. (Incoherent module) Let C be an unsatisfiable class in an ontology O. An incoherent module $\mathcal{M}$ for O and C is a subset of O such that $\mathcal{M} \models C \sqsubseteq \bot \Leftrightarrow O \models C \sqsubseteq \bot$.

Definition 6. (Syntactic locality) Let $\mathcal{S}$ be a signature, R a role, and C a class. Let $A^{\bot} \notin \mathcal{S}$ be an atomic class and let $R^{\bot} \notin Rol(\mathcal{S})$ be a role. Two sets of classes, namely, $C^{\top}_{\mathcal{S}}$ and $C^{\bot}_{\mathcal{S}}$, are recursively defined by the following rules:
$C^{\top}_{\mathcal{S}} ::= (\neg C^{\bot})|(C^{\top}_1 \sqcap C^{\top}_2)$,
$C^{\bot}_{\mathcal{S}} ::= A^{\bot}|(\neg C^{\top})|(C \sqcap C^{\bot})|(\exists R^{\bot}.C)|(\exists R.C^{\bot})|(\geqslant nR^{\bot}.C)|(\geqslant nR.C^{\bot})$,
where $C^{\bot} \in C^{\bot}_{\mathcal{S}}$, $C^{\top}_{(i)} \in C^{\top}_{\mathcal{S}}, i = 1, 2$.

An axiom is syntactically local with respect to $\mathcal{S}$ if it is of one of the following forms:
(1) $R^{\bot} \sqsubseteq R$, (2) $\mathrm{Trans}(R^{\bot})$, (3) $C\bot \sqsubseteq C$, (4) $C \sqsubseteq C^{\top}$.

An OWL DL ontology O is syntactically local with respect to $\mathcal{S}$ if all axioms in O are syntactically local with respect to $\mathcal{S}$.

Classes in $C^{\bot}_{\mathcal{S}}$ become equivalent to the bottom class $\bot$ if $A^{\bot}$ or $R^{\bot}$ not in S is replaced with $\bot$ or $\varnothing$. Similarly, Classes in $C^{\top}_{\mathcal{S}}$ become equivalent to the top class $\top$ under the conditions of this replacement. After these replacements, syntactically local axioms become tautologies.

Proposition 1 (Testing locality) [35] Let $\mathcal{S}$ be a signature, C a class and α an axiom; then the localities of C and α for $\mathcal{S}$ can be defined recursively as follows:

$\tau(C, \mathcal{S})$ ::=	$\tau(\top, \mathcal{S})$	$= \top$	(a)
	$\tau(A, \mathcal{S})$	$= \bot$ if $A \notin \mathcal{S}$ and otherwise $= A$	(b)
	$\tau(\{a\}, \mathcal{S})$	$= a$	(c)
	$\tau(C_1 \sqcap C_2, \mathcal{S})$	$= \tau(C_1, \mathcal{S}) \sqcap \tau(C_2, \mathcal{S})$	(d)
	$\tau(\neg C_1, \mathcal{S})$	$= \neg\tau(C_1, \mathcal{S})$	(e)
	$\tau(\exists R.C_1, \mathcal{S})$	$= \bot$ if $\mathrm{Sig}(R) \nsubseteq \mathcal{S}$ and otherwise $= \exists R.\tau(C_1, \mathcal{S})$	(f)
	$\tau(\geq nR.C_1, \mathcal{S})$	$= \bot$ if $\mathrm{Sig}(R) \nsubseteq \mathcal{S}$ and otherwise $= \geq nR.\tau(C_1, \mathcal{S})$	(g)
$\tau(\alpha, \mathcal{S})$ ::=	$\tau(C_1 \sqsubseteq C_2, \mathcal{S})$	$= (\tau(C_1, \mathcal{S}) \sqsubseteq \tau(C_2, \mathcal{S}))$	(h)
	$\tau(R_1 \sqsubseteq R_2, \mathcal{S})$	$= \bot \sqsubseteq \bot$ if $\mathrm{Sig}(R_1) \nsubseteq \mathcal{S}$, otherwise	
		$= \exists R_1.\top \sqsubseteq \bot$ if $\mathrm{Sig}(R_2) \nsubseteq \mathcal{S}$, otherwise$= (R_1 \sqsubseteq R_2)$	(i)
	$\tau(a : C, \mathcal{S})$	$= a : \tau(C, \mathcal{S})$	(j)
	$\tau(R(a, b), \mathcal{S})$	$\top \sqsubseteq \bot$ if $R \notin \mathcal{S}$ and otherwise $= R(a, b)$	(k)
	$\tau(\mathsf{Trans}(R), \mathcal{S})$	$= \bot \sqsubseteq \bot$ if $R \notin \mathcal{S}$ and otherwise $= \mathsf{Trans}(R)$	(l)
	$\tau(\mathsf{Funct}(R), \mathcal{S})$	$= \bot \sqsubseteq \bot$ if $\mathrm{Sig}(R) \notin \mathcal{S}$ and otherwise $= \mathsf{Funct}(R)$	(m)
$\tau(O, \mathcal{S})$ ::=	$\bigcup_{\alpha \in O} \tau(\alpha, \mathcal{S})$		(n)

The following example shown in Figure 4 from [35] will suffice to illustrate the point.

In Figure 4(1), let $O = \{Genetic\_Fibrosis \equiv Fibrosis \sqcap \exists has\_Origin.Genetic\_Origin\}$ and $\mathcal{S}_1 = \{Fibrosis, Genetic\_Origin\}$. Firstly, $\tau(Genetic\_Fibrosis, \mathcal{S}_1) = \bot$ because $Genetic\_Fibrosis \notin \mathcal{S}_1$ according to the case (b) from Proposition 1. Therefore, the left hand side of the axiom $Genetic\_Fibrosis \equiv Fibrosis \sqcap \exists has\_Origin.Genetic\_Origin$ is replaced by $\bot$. That is, $\bot \equiv Fibrosis \sqcap \exists has\_Origin.Genetic\_Origin$. Secondly, $\tau(has\_Origin, \mathcal{S}_1) = \bot$ because $Sig(has\_Origin) \nsubseteq \mathcal{S}_1$ according to the case (f) from Proposition 1. Therefore, the part $\exists has\_Origin.Genetic\_Origin$ on the right-hand side of the axiom is replaced by $\bot$. That is, $\bot \equiv Fibrosis \sqcap \bot$. We know that $\bot \equiv Fibrosis \sqcap \bot$

is a tautology. Hence $\mathcal{O}$ is local with respect to $\mathcal{S}_1$.

In Figure 4(2), let $\mathcal{O} = \{Genetic\_Fibrosis \equiv Fibrosis \sqcap \exists has\_Origin.Genetic\_Origin\}$ and $\mathcal{S}_2 = \{Genetic\_Fibrosis, has\_Origin\}$. $\tau(Fibrosis, \mathcal{S}_2) = \bot$ because $Fibrosis \notin \mathcal{S}_2$ according to the case (b) from Proposition 1. Therefore, the part $Fibrosis$ on the right hand side of the axiom is replaced by $\bot$. Moreover, $\tau(Genetic\_Origin, \mathcal{S}_2) = \bot$ because $Genetic\_Origin \notin \mathcal{S}_2$ according to the case (b) from Proposition 1. Therefore, the part $Genetic\_Origin$ is also replaced by $\bot$. That is, $Genetic\_Fibrosis \equiv \bot \sqcap \exists has\_Origin.\bot \Leftrightarrow Genetic\_Fibrosis \equiv \bot$. We know that $Genetic\_Fibrosis \equiv \bot$ is not a tautology. Hence $\mathcal{O}$ is not local with respect to $\mathcal{S}_2$.

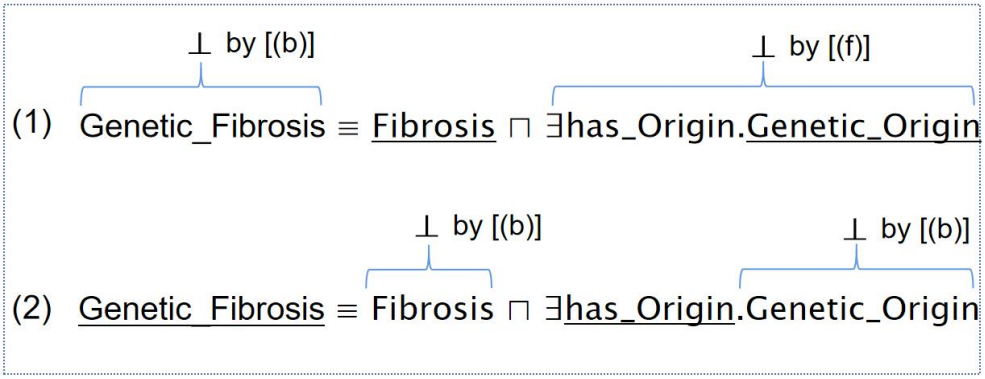


Figure 4. Sketch of testing localities for given signatures.

We modify the algorithm presented in [14] to extract a module related to a signature, and implement the following Algorithm 2 for extracting the module for an unsatisfiable class.

Algorithm 2: ExtractModule($\mathcal{O}$,C).

Input: $\mathcal{O}$:an ontology;
C:an unsatisfiable class
Output: $\mathcal{M}_C$: C-module of $\mathcal{O}$

```
1  M_C ← ∅, O' ← O
2  while O'! = ∅ do
3     α ← SelectAxiom(O')
4     if LocalityTest(α,{C}∪Sig(M_C)) then
5        O' ← O' \ {α}              # α is processed
6     else
7        M_C ← M_C ∪ {α}            # move α into M_C
8        O' ← O \ M_C               # reset O' to the complement of M_C
9     end if
10 end while
11 return M_C
```

Given an ontology $\mathcal{O}$ and an unsatisfiable class C, Algorithm 2 retrieves a fragment $\mathcal{M}_C \subseteq \mathcal{O}$ as follows. First, $\mathcal{M}_C$ is initialized to $\emptyset$, and the ontology $\mathcal{O}$ is copied to $\mathcal{O}'$ (line 1). Second, an axiom α is randomly selected from $\mathcal{O}'$ (line 3). If α is locality with respect to the union of C and the signature

of $\mathcal{M}_C$, then α is removed from $\mathcal{O}'$ (lines 4–5). Otherwise, α is added to $\mathcal{M}_C$ (line 7). Lastly, $\mathcal{M}_C$ is removed from $\mathcal{O}$ and the remaining subset is copied thereafter to $\mathcal{O}'$ (line 8). These steps are repeated until $\mathcal{O}' = \emptyset$.

For example, considering the following ontology $\mathcal{O}_2$ presented in Reference [14].

α_1	Cystic-Fibrosis ≡ Fibrosis ⊓∃ located-In.Pancreas ⊓∃ has-Origin.Genetic-Origin
α_2	Genetic-Fibrosis ≡ Fibrosis ⊓∃ has-Origin.Genetic-Origin
α_3	Fibrosis ⊓∃ located-In.Pancreas ⊑ Genetic-Fibrosis
α_4	Genetic-Fibrosis ⊑ Genetic-Disorder
α_5	DEFBI-Gene ⊑ Immuno-Protein-Gene ⊓∃ associated-With.Cystic-Fibrosis

The trace of Algorithm 2 for $\mathcal{O}_2 = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ and S=$\{Cystic-Fibrosis, Genetic-Disorder\}$ is described in Table 2.

Table 2. Trace of Algorithm 2 for $\mathcal{O}_2$ and S.

♯	$\mathcal{M}_C$	$\mathcal{O}'_2$	New elements in S ∪ Sig($\mathcal{M}_C$)	α	loc.?
1	$\emptyset$	$\alpha_1 - \alpha_5$	Cystic-Fibrosis, Genetic-Disorder	α_1	No
2	α_1	$\alpha_2 - \alpha_5$	Fibrosis, located-In, Pancreas, has-Origin, Genetic-Origin	α_2	No
3	α_1, α_2	$\alpha_3 - \alpha_5$	Genetic-Fibrosis	α_3	No
4	$\alpha_1 - \alpha_3$	α_4, α_5	—	α_4	No
5	$\alpha_1 - \alpha_4$	α_5	—	α_5	Yes
6	$\alpha_1 - \alpha_4$	—	—	—	

Theorem 1 guarantees the correctness of Algorithm 2.

Theorem 1 (Correctness of Algorithm 2). For any unsatisfiable class C in an ontology $\mathcal{O}$, Algorithm 2 returns a C-module of $\mathcal{O}$.

Proof. (1) Algorithm 2 terminates for any input C in $\mathcal{O}$.

In every iteration of the while loop, either the size of $\mathcal{M}_C$ increases, or the size of $\mathcal{M}_C$ remains the same as the size of $\mathcal{O}'$ decreases. This means that Algorithm 2 terminates in quadratic time in the number of axioms in $\mathcal{O}$, assuming a constant time locality test.

(2) The output $\mathcal{M}_C$ of Algorithm 2 is a locality-based C-module in $\mathcal{O}$.

Given that α can appear in line 3 of the algorithm, α is local with respect to $\{C\} \cup Sig(\mathcal{M}_C)$ if α is neither in $\mathcal{M}_C$ nor in $\mathcal{O}$'. Moreover, α remains in $\mathcal{O} \setminus (\mathcal{M}_C \cup \mathcal{O})$ if $\{C\} \cup Sig(\mathcal{M}_C)$ does not change. □

3.2. DOBP algorithm

In Reference [36], the authors proposed three heuristic strategies for the unsatisfiability of classes.

(1) Local unsatisfiability. The combination of direct restrictions and superclasses is unsatisfiable.

(2) Propagated unsatisfiability. The combination of direct restrictions and superclasses are unsatisfiable except that some classes used in them are unsatisfiable.

(3) Global unsatisfiability. The domain/range constraints, which along with other information can be used to infer that the class is unsatisfiable.

On the basis of the above three strategies, Ji proposed a novel debugging algorithm in Reference [37] presenting four types of incoherent patterns as follows.

(1) Isa-Disjoint pattern: $X \sqsubseteq Y, X \sqsubseteq Z, Y \sqsubseteq \neg Z$

The pattern indicates that the superclasses Y and Z of X are disjointed.

(2) Exist-Bottom pattern: $X \sqsubseteq \exists r.Y, Y \sqsubseteq \bot$

The pattern indicates that the filler Y of existential restriction is a bottom class, that is, the semantic of Y is empty.

(3) Exist-All pattern: $X \sqsubseteq \exists r.Y, X \sqsubseteq \forall r.Z, Y \sqsubseteq \neg Z$

(4) Exist-Domain pattern: $X \sqsubseteq \exists r.Y, X \sqsubseteq \neg Z$, domain$(\exists r.Y = Z)$

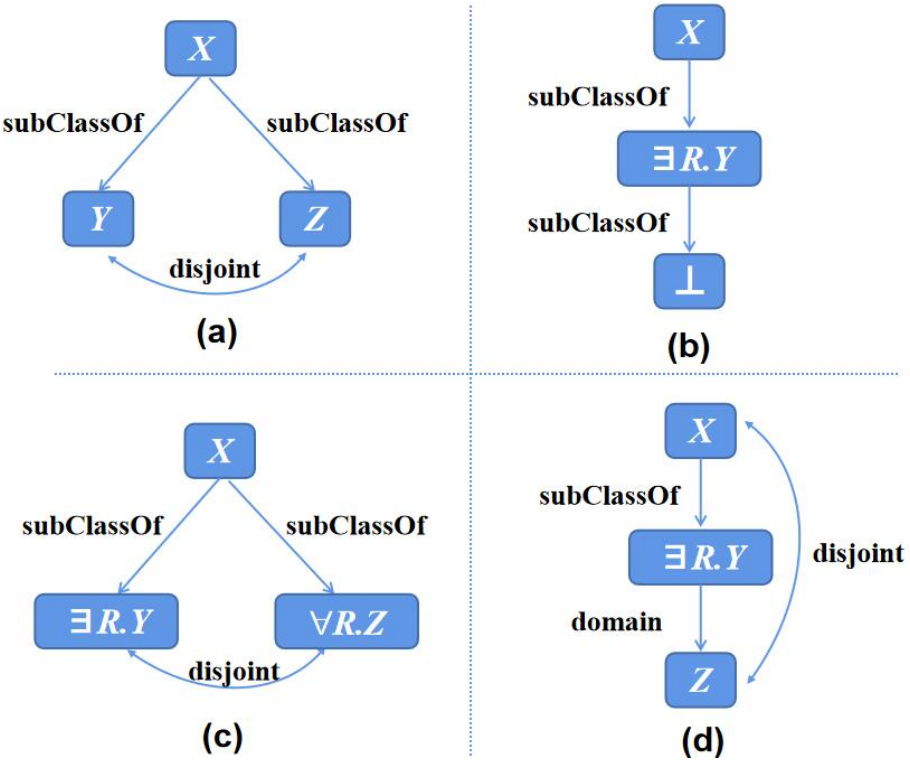


Figure 5. Four types of incoherent patterns.

Figure 5 shows the incoherent patterns proposed in Reference [37]. Each incoherent pattern can be represented as a directed graph, where the vertex indicates class and the arc from the vertex A to the vertex B indicates a relation, corresponding to a logical axiom, between A and B. For example, $< X, \exists r.Y >$ denotes the arc from X to $\exists r.Y$ and it corresponds to the axiom $X \sqsubseteq \exists r.Y$.

Patterns provide beneficial information to explain unsatisfiability according to the structure of an ontology. Various patterns are also presented in Reference [38], which lists the following antipatterns.

(1) AntiPattern AndIsOr (AIO): $C_1 \sqsubseteq \exists R.(C_2 \sqcap C_3), Disj(C_2, C_3)$.

(2) AntiPattern OnlynessIsLoneliness (OIL): $C_1 \sqsubseteq \forall R.C_2, C_1 \sqsubseteq \forall R.C_3, Disj(C_2, C_3)$

(3) AntiPatterns UniversalExistence (UE): $C_1 \sqsubseteq \exists R.C_2, C_1 \sqsubseteq \forall R.C_3, Disj(C_2, C_3)$

(4) AntiPattern UniversalExistenceWithInverseProperty (UEWIP): $C_2 \sqsubseteq \exists R^{-}.C_1, C_1 \sqsubseteq \forall R.C_3, Disj(C_2, C_3)$.

(5) AntiPattern EquivalenceIsDifference (EID) $C_1 \equiv C_2, Disj(C_1, C_2)$.

Algorithm 3: DOBP($\mathcal{O}$,C).

Input: $\mathcal{O}$:ontology; C:an unsatisfiable class
Output: MinAs of C

```
1  CS = ∅
2  if C ∈ CS then
3    return null
4  CS ← CS ∪ {C}
5  MinAs ← DOBP_IsaDisjoint(O, C)
6  MinAs ← MinAs ∪ DOBP_ExistBot(O, C)
7  MinAs ← MinAs ∪ DOBP_ExistAll(O, C)
8  MinAs ← MinAs ∪ DOBP_ExistDomain(O, C)
9  return MinAs
```

For each pattern, Algorithm 3 first finds the disjoint relations and searches a set of paths thereafter in connection with the relations. For example, finding disjoint relations between A and B involves iterating the ancestors of A and those of B. If one ancestor of A is disjoint with the other ancestor of B, then A is also disjoint with B [37].

For example, considering the following $\mathcal{O}_3$:

$C1 \sqsubseteq \exists R1.(C2 \sqcap \exists R2.C3)$,

$C3 \sqsubseteq \exists R3.C4 \sqcap \forall R3.C5$,

$C5 \sqsubseteq C6$,

$C4 \sqsubseteq \neg C6$.

In $\mathcal{O}_3$, $C1$ and $C3$ are unsatisfiable. Now, $\mathcal{O}_3$ is normalized as follows:

$C1 \sqsubseteq \exists R1.X$,

$X \sqsubseteq \exists R2.C3$,

$X \sqsubseteq C2$,

$C3 \sqsubseteq \exists R3.C4$,

$C3 \sqsubseteq \forall R3.C5$,

$C5 \sqsubseteq C6$,

$C4 \sqsubseteq \neg C6$.

Here, X is a new added concept. Then, X is also unsatisfiable because $X \sqsubseteq \exists R2.C3$ and $C3$ is unsatisfiable

For $C1$ in $\mathcal{O}_3$, the Exist-Bottom pattern shown in Figure 5(b) is detected as $C1 \sqsubseteq \exists R1.X$. To explain why X is unsatisfiable, DOBP needs to be invoked again. Then the Exist-Bottom pattern is detected again for X because $X \sqsubseteq \exists R2.C3$.

For $C3$, the Exist-All pattern shown in Figure 5(c) is detected because $\exists R3.C4$ and $\forall R3.C5$ share $R3$ and $\langle C4, \neg C6\rangle$ is a disjoint relation. For this relation, we have $\{\langle C4, C4\rangle, \langle C5, C6\rangle, \langle C3, C3\rangle\}$. After expanding the pairs, we obtain $\{\{C5, C6\}\}$. Then we have $\{C3 \sqsubseteq \exists R3.C4 \sqcap \forall R3.C5, C5 \sqsubseteq C6, C4 \sqsubseteq \neg C6\}$ that is the MinAs of $C3$.

3.3. Module-DOBP algorithm

The performance of the DOBP algorithm is significantly affected if the size of the ontology and the number of unsatisfiable classes are large. However, the modularization method can extract a sub-ontology as needed. From this perspective, we combine the modularization method with the DOBP algorithm and propose the Module-DOBP algorithm (see Algorithm 4).

In lines 3 to 4, the algorithm extracts the module of the unsatisfiable class C by calling the Extract-Module sub-routing (Algorithm 2), and adds the module to the results set $\mathcal{M}$. For each unsatisfiable class C in CS, the algorithm computes the MinAs related to the four patterns based on the extracted module $\mathcal{M}$ and C in lines 7 to 10.

Algorithm 4: Module-DOBP($\mathcal{O}$).

Input: $\mathcal{O}$:ontology
Output: MinAs

```
1  CS ← O, M = ∅
2  for C in CS
3      M_C ←ExtractModule(O, C)
4      M = M ∪ M_C
5  end for
6  for C in CS
7      MinAs ← DOBP_IsaDisjoint(M, C)
8      MinAs ← MinAs ∪ DOBP_ExistBot(M, C)
9      MinAs ← MinAs ∪ DOBP_ExistAll(M, C)
10     MinAs ← MinAs ∪ DOBP_ExistDomain(M, C)
11 end for
12 return MinAs
```

Theorem 2. Let $\mathcal{M}$ be the incoherent module of an incoherent ontology $\mathcal{O}$. For any unsatisfiable class C, let MinA($\mathcal{O}, C$) be MinA of $\mathcal{O}$. Thereafter, MinA($\mathcal{M}, C$) exists such that MinA($\mathcal{M}, C$)=MinA($\mathcal{O}, C$), where MinA($\mathcal{M}, C$) is the MinA of $\mathcal{M}$.

Proof. We prove on the basis of the expansion-contraction process of Algorithm 1.

Let $\mathcal{O} = \mathcal{M} \cup \mathcal{N}$, and $\mathcal{M} \cap \mathcal{N} = \varnothing$. According to Definition 4, we have $\mathcal{M} \models C \sqsubseteq \bot \Leftrightarrow \mathcal{O} \models C \sqsubseteq \bot$.

1) Expansion process of Algorithm 1.

Let $S_{\mathcal{O}} = \varnothing$ and $S_{\mathcal{M}} = \varnothing$ be the expansion set of $\mathcal{O}$ and $\mathcal{M}$. We obtain $S_{\mathcal{O}} = \{\alpha_1\}$ after the first axiom α_1 is added to $S_{\mathcal{O}}$. We consider two cases:

(1) $\alpha_1 \in \mathcal{N}$. Considering $\mathcal{M} \cap \mathcal{N} = \varnothing$, we have $\alpha_1 \notin \mathcal{M}$. Then, $S_{\mathcal{M}} = \varnothing$.

(2) $\alpha_1 \in \mathcal{M}$. Incorporating α_1 into $\mathcal{M}$ we have $S_{\mathcal{M}} = \{\alpha_1\}$. After incorporating k axioms into $\mathcal{O}$ we have $S_{\mathcal{O}} = \{\alpha_1, ..., \alpha_k\} \models C \sqsubseteq \bot$. Meanwhile, the set of axioms $\{\alpha_1, ..., \alpha_k\}$ are also incorporated into $\mathcal{M}$, and we have $S_{\mathcal{M}} = \{\alpha_1, ..., \alpha_k\}$. The expansion process ends and we have $\mathcal{M} \models C \sqsubseteq \bot \Leftrightarrow \mathcal{O} \models C \sqsubseteq \bot$, $S_{\mathcal{M}} \subseteq S_{\mathcal{O}}$.

2) Contraction process of Algorithm 1.

Given $S_{\mathcal{M}} \subseteq S_{\mathcal{O}}$, we let $S_{\mathcal{O}} = S_{\mathcal{M}} \cup S_{\mathcal{N}}$, $S_{\mathcal{M}} \cap S_{\mathcal{N}} = \varnothing$. Given $S_{\mathcal{O}} \models C \sqsubseteq \bot \Leftrightarrow S_{\mathcal{M}} \models C \sqsubseteq \bot$ we have $S_{\mathcal{N}} \not\models C \sqsubseteq \bot$. Let $S_{\mathcal{O}}$ be the set after removing the axiom α_x from $S_{\mathcal{O}}$. Two cases exist as follows.

(1) $\alpha_x \in S_{\mathcal{M}}$. Let $S_{\mathcal{M}}$ be the axiom set after removing α_x from $S_{\mathcal{M}}$. We consider the following two cases.

(a) $S_{\mathcal{O}} \models C \sqsubseteq \bot$. It indicates that a_x is not responsible for the unsatisfiability of C. Thus, $S_{\mathcal{M}} \models C \sqsubseteq \bot$.

(b) $S_{\mathcal{O}} \not\models C \sqsubseteq \bot$. It indicates that a_x is responsible for the unsatisfiability of C. Thus, the unsatisfiable C becomes satisfiable. In this case, we must reinsert a_x into $S_{\mathcal{O}}$ and $S_{\mathcal{M}}$; thereafter, we have $S_{\mathcal{O}} = S_{\mathcal{O}} \cup \{a_x\}$ and $S'_{\mathcal{M}} = S'_{\mathcal{M}} \cup \{a_x\}$.

(2) $a_x \in S_{\mathcal{N}}$. Given that $S_{\mathcal{M}} \cup S_{\mathcal{N}} = \varnothing$, we have $a_x \notin S_{\mathcal{M}}$. In such a case, $S_{\mathcal{M}} \models C \sqsubseteq \bot$. Considering $S_{\mathcal{P}} \not\models C \sqsubseteq \bot$, we have $S_{\mathcal{M}} \models C \sqsubseteq \bot$.

After all axioms in $\mathcal{O}$ are tested by following the preceding steps, we have $S_{\mathcal{O}} = S_{\mathcal{M}}$.

On the basis of the preceding two processes, we have $\mathrm{MinA}(\mathcal{O}, C)=S_{\mathcal{O}}$ and $\mathrm{MinA}(\mathcal{M}, C)=S_{\mathcal{M}}$. Therefore, $\mathrm{MinA}(\mathcal{O}, C)=\mathrm{MinA}(\mathcal{M}, C)$. □

Theorem 2 indicates that for any unsatisfiable class C, we can obtain the incoherent module $\mathcal{M}_C$ responsible for the unsatisfiability of C such that $\mathrm{MinA}(\mathcal{O}, C)=\mathrm{MinA}(\mathcal{M}, C)$.

Theorem 3 (correctness of Algorithm 4). Let $\mathcal{M}$ be the incoherent module of an incoherent ontology $\mathcal{O}$. For any unsatisfiable class C, we have $\mathrm{MinAs}(\mathcal{M}, C)=\mathrm{MinAs}(\mathcal{O}, C)$.

Proof. Let $\mathrm{MinAs}(\mathcal{O}, C)=\bigcup_1^m \mathrm{MinA}(\mathcal{O}, C)_i$, where m is the number of $\mathrm{MinA}(\mathcal{O}, C)$. For any $\mathrm{MinA}(\mathcal{O}, C)_i \in \mathrm{MinAs}(\mathcal{O}, C)$, according to Theorem 2, $\mathrm{MinA}(\mathcal{M}, C)_i$ exists such that $\mathrm{MinA}(\mathcal{O}, C)_i=\mathrm{MinA}(\mathcal{M}, C)_i$. Therefore, we have $\mathrm{MinAs}(\mathcal{M}, C)=\bigcup_1^m \mathrm{MinA}(\mathcal{M}, C)_i=\bigcup_1^m \mathrm{MinA}(\mathcal{O}, C)_i=\mathrm{MinAs}(\mathcal{O}, C)$. □

4. Experiment

Experimental evaluations were performed on a laptop with a 1.60 GHz Intel Core i5 CPU and 16 GB of main memory. We conducted all experiments using Pellet 2.3.1* as a reasoner for satisfiability tests and OWL API 3.4.3 † for ontology loading and manipulation. The experimental corpus, source codes and experimental results are available at http://www.zhyweb.cn/mradon/index.html.

4.1. Benchmark ontologies

The benchmark ontologies used in our experiments were taken from the Open Biological Ontology library (see http://krr-nas.cs.ox.ac.uk/ontologies/lib/OBO/). These ontologies are well-known in the life sciences because they are often used in real-world applications. We collected 15 biological ontologies (see Table 2) as benchmarks for our experimental evaluations. The test ontologies are complex because they use numerous OWL DL constructors and some of them include a large number of axioms. To obtain the incoherent ontologies, we adopted the "incoherent-generating" method given in Reference [12]. For example, if $\mathcal{O} = \{A \sqsubseteq B, A \sqsubseteq C\}$, then $\mathcal{O}$ is evidently coherent. We applied four classes A, B, C and D. Then we created a pair of complementary classes by randomly selecting two classes. For example, $A \sqsubseteq \neg B$, $A \sqsubseteq \neg C$, $A \sqsubseteq C$, $B \sqsubseteq C$, $B \sqsubseteq D$ or $C \sqsubseteq D$. There were a total of six possibilities. More details can be found in Table 3.

*Pellet is available at https://github.com/stardog-union/pellet

†https://sourceforge.net/projects/owlapi/

In Table 3, the first two columns indicate the IDs and names of the 15 ontologies. The third column represents the expressivity. The expressivities of the test ontologies ranges from $\mathcal{ALCH}$ through to $\mathcal{SHOIF}(\mathcal{D})$. The next four columns refer to number of classes ($\sharp C$), number of properties ($\sharp R$), the number of axioms ($\sharp onto.$) and the number of unsatisfiable classes ($\sharp u.c.$), respectively.

Table 3. Characteristics of test ontologies used in experiments.

ID.	Name	Exp.	$\sharp C$	$\sharp P$	$\sharp onto.$	$\sharp u.c.$
O_1	NIF_Dysfunction	$\mathcal{SHOIF}(\mathcal{D})$	2749	60	3511	25
O_2	amphibian_anatomy	$\mathcal{SH}$	700	2	708	14
O_3	cell	$\mathcal{ALCH}$	814	32	645	13
O_4	cellular_component	$\mathcal{ALCH}$	1111	32	761	34
O_5	brenda	$\mathcal{ALCH}$	3138	3	3957	11
O_6	Cellular09	$\mathcal{SH}$	2370	4	4552	20
O_7	Cellular12	$\mathcal{SR}$	3121	7	5818	12
O_8	cereal	$\mathcal{ALCH}$	869	2	1433	12
O_9	cereal_anatomy	$\mathcal{SR}$	1271	4	2281	16
O_{10}	envo	$\mathcal{SH}$	1226	3	1445	18
O_{11}	envo_xp	$\mathcal{SH}$	1779	8	2218	30
O_{12}	event	$\mathcal{SH}$	3829	4	7380	31
O_{13}	fix	$\mathcal{ALCH}$	1163	2	1784	19
O_{14}	fly	$\mathcal{SHI}$	6322	3	11014	12
O_{15}	fly_anatomy	$\mathcal{SRI}$	7798	21	19574	15

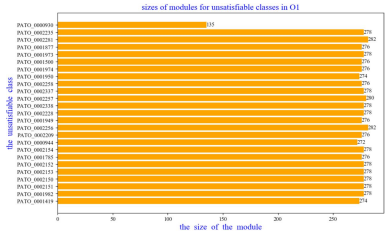
4.2. Experimental analysis

For every incoherent ontology, and for each unsatisfiable class in the signature, we extracted the corresponding incoherent modules using Algorithm 2.

Table 4 presents the maximum, minimum and average sizes of the modules and the standard deviation. Among the 15 ontologies, O_{15} had the largest maximum, minimum and average module sizes because O_{15} had an extremely complex ontology structure with 7798 classes and 19574 axioms. However, MAX in O_3 only contained 6 axioms because it has a markedly simple with 814 classes and 645 axioms.

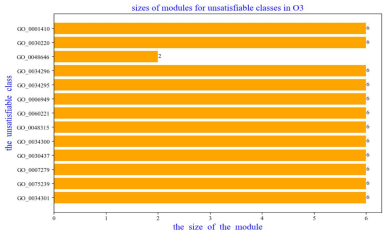
Table 4. Maximum/minimum/average of modules for unsatisfiable classes.

ID.	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
MAX	282	18	6	15	26	151	21	53	60	48	90	42	93	142	585
MIN	135	14	2	6	24	28	11	30	22	12	21	19	25	138	342
AVG	271.6	17.6	5.7	11.4	25.8	86.1	16.6	45.2	45.5	29.1	53.2	33.3	53.4	141.2	467.9
STDEV	28.0	1.1	1.1	1.6	0.6	34.4	3.0	7.2	10.4	9.6	18.6	3.8	22.3	1.3	116.0

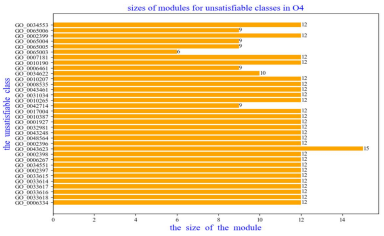


(a) sizes of modules for unsatisfiable classes in O_1

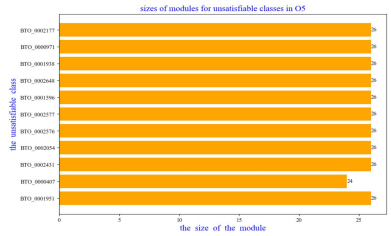
(b) sizes of modules for unsatisfiable classes in O_2



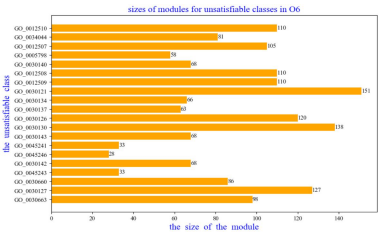
(c) sizes of modules for unsatisfiable classes in O_3



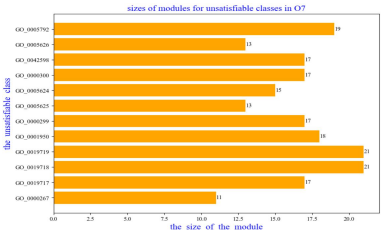
(d) sizes of modules for unsatisfiable classes in O_4



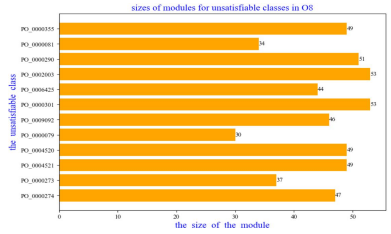
(e) sizes of modules for unsatisfiable classes in O_5



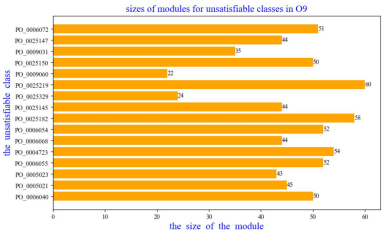
(f) sizes of modules for unsatisfiable classes in O_6



(g) sizes of modules for unsatisfiable classes in O_7

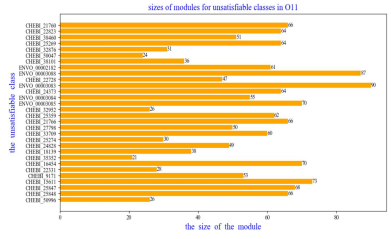


(h) sizes of modules for unsatisfiable classes in O_8

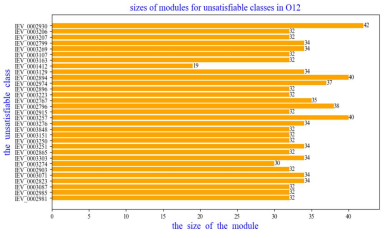


(i) sizes of modules for unsatisfiable classes in O_9

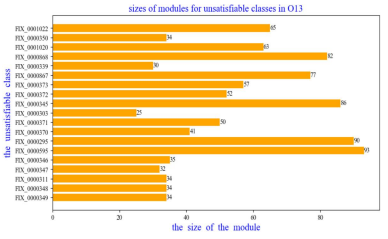
(j) sizes of modules for unsatisfiable classes in O_{10}



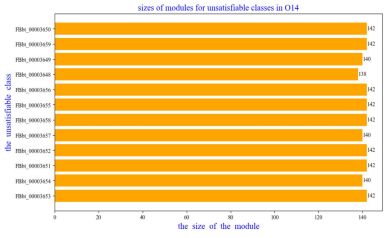
(k) sizes of modules for unsatisfiable classes in O_{11}



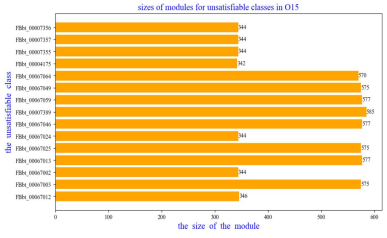
(l) sizes of modules for unsatisfiable classes in O_{12}



(m) sizes of modules for unsatisfiable classes in O_{13}

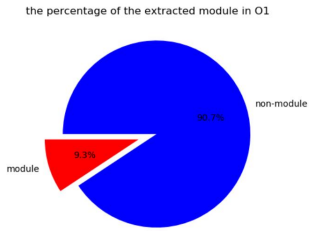


(n) sizes of modules for unsatisfiable classes in O_{14}

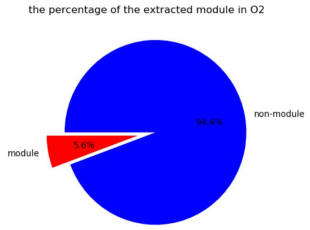


(o) sizes of modules for unsatisfiable classes in O_{15}

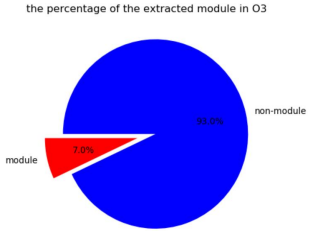
Figure 6. Sizes of modules for the unsatisfiable classes in the 15 ontologies.



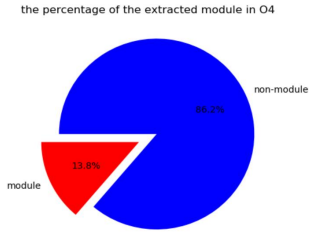
(a) the percentage of the module vs. the non-module for O_1



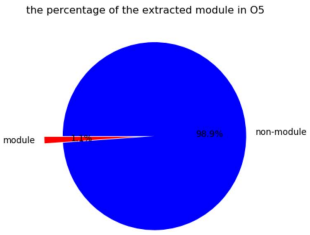
(b) the percentage of the module vs. the non-module for O_2



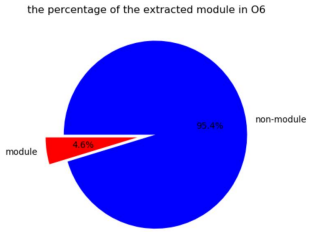
(c) the percentage of the module vs. the non-module for O_3



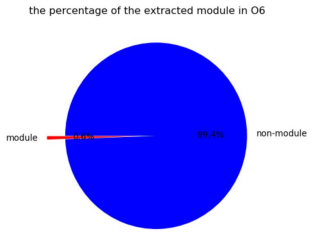
(d) the percentage of the module vs. the non-module for O_4



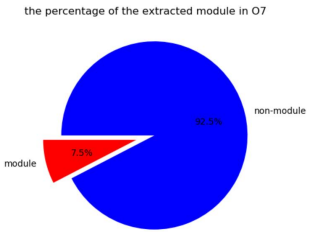
(e) the percentage of the module vs. the non-module for O_5



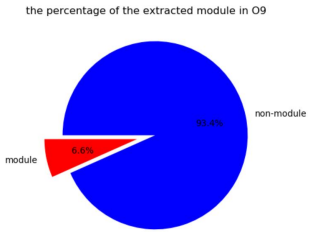
(f) the percentage of the module vs. the non-module for O_6



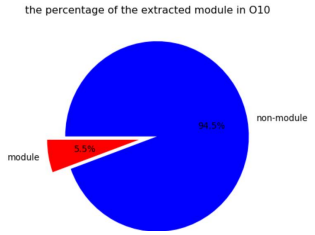
(g) the percentage of the module vs. the non-module for O_7



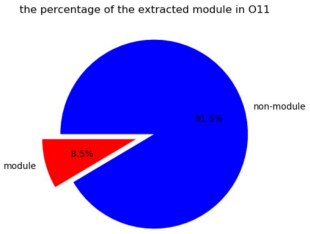
(h) the percentage of the module vs. the non-module for O_8



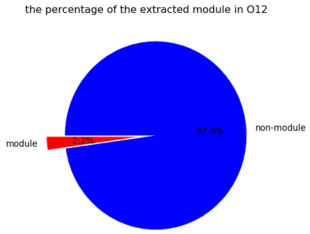
(i) the percentage of the module vs. the non-module for O_9



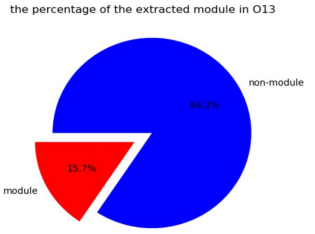
(j) the percentage of the module vs. the non-module for O_{10}



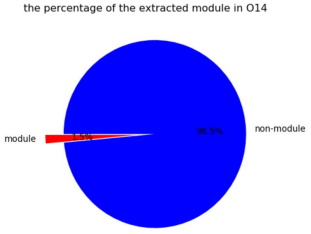
(k) the percentage of the module vs. the non-module for O_{11}



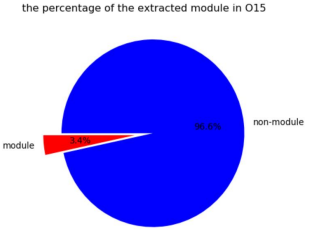
(l) the percentage of the module vs. the non-module for O_{12}



(m) the percentage of the module vs. the non-module for O_{13}



(n) the percentage of the module vs. the non-module for O_{14}



(o) the percentage of the module vs. the non-module for O_{15}

Figure 7. Percentages of incoherent modules vs. non-incoherent modules for the 15 ontologies.

Figure 6 shows the sizes of the modules for each unsatisfiable class in 15 ontologies; the Y-axis represents the unsatisfiable class and the X-axis represents the size of the corresponding module.

Apart from *PATO_0000930*, all module sizes for the 24 unsatisfiable classes were found to be equally likely in O_1. The average size was close to the maximum. In O_2, the modules were the same size (18), except the two classes *AAO_0000679* and *AAO_0000199*. All but one of the modules had the same size, which also occurred in O_3 and O_5. In O_4, the unsatisfiable class GO-0043623 had the largest module, GO-0065003 had the smallest module and the size for 76% the modules was 12. A total of 10 out of 11 unsatisfiable classes in O_5 had the same sized module. The sizes of modules for all unsatisfiable classes in O_{14} were relatively close to each other.

The percentages of extracted incoherent modules vs. non-incoherent modules for 15 ontologies are summarized in Figure 7.

For all incoherent ontologies, we obtained extremely small modules. The largest module obtained for O_{13} was only 15.7%. This result indicates that the dependencies between the different classes are extremely strong, and that structures of the ontology are more complex than other ontologies.

The following observations were obtained on the basis of the above-mentioned 15 charts:

(1) The size of the module is smaller than that of the non-module. For example, the percentage of the module in O_7 was found to be 0.6%. This means that only 34 axioms out of the 5818 axioms are responsible for the incoherence.

(2) The percentage of the module in O_{13} was found to be the highest among all ontologies, although it was below 16%.

(3) The modules extracted using our Module-DOBP algorithm were found to be significantly smaller than the sizes of the original ontologies. For example, the module $\mathcal{M}_6$ obtained was 6/1000 of the size of O_6.

Table 5 shows the comparison results for Module-DOBP and DOBP. $\mathcal{M}$ represents the size of the extracted module (i.e., the number of axioms in the module). T_M denotes the time required to extract the module. T_{DOBP} displays the time to compute MinAs based on the extracted modules. The Module-DOBP column shows the total time to extract the module and compute the MinAs (i.e., $T_M + T_{DOBP}$). The DOBP column show the time to compute the MinAs based on the original ontology.

For the majority of the ontologies (13/15) in Table 4, the time required to extract modules was below 5 seconds and all 15 ontologies required less than 10 seconds. O_{15} required the most time to extract the modules because it had the axioms (19574).

We found that Module-DOBP performed better than DOBP for the majority of cases. Module-DOBP is considerably efficient because it was designed to extract incoherent modules and compute MinAs based on the module. Based on the results for O_7, O_{12}, O_{14} and O_{15}, it turns out that module-extracting optimization is necessary to make our Module-DOBP perform well on large ontologies. For example, the size of O_{15} was 19574. The runtime of Module-DOBP was 12.819. However, the runtime of DOBP was 140 times longer than that for Module-DOBP. DOBP performed inefficiently because it worked on the entire ontology. Take O_{15} as an example. Module-DOBP took 12.819 seconds to complete the computation. However, DOBP could not complete the task within 30 minutes. Module-DOBP performed worse for O_{13} than DOBP. The reason is that in such a case, the additional time to extract the module is relatively large compared with the total time.

The runtime performance of Module-DOBP was strongly affected by the time consumed by extracting modules. When more time was needed, performance was substantially degraded.

Table 5. Comparison results for Module-DOBP and DOBP.

ID.	♯*onto*.	♯*u.c*.	$\mathcal{M}$	T_M	T_{DOBP}	Module-DOBP	DOBP	♯*MinAs*
$\mathcal{O}_1$	3511	25	327	3.995	2.151	6.146	20.579	25
$\mathcal{O}_2$	708	14	40	1.766	0.79	2.556	4.475	14
$\mathcal{O}_3$	645	13	45	2.302	0.607	**2.909**	**1.595**	13
$\mathcal{O}_4$	761	34	105	2.486	1.441	3.927	9.042	34
$\mathcal{O}_5$	3957	11	44	3.006	0.505	**3.511**	**2.728**	11
$\mathcal{O}_6$	4552	20	210	4.41	7.329	11.739	612.6	119
$\mathcal{O}_7$	5818	12	34	3.112	0.72	3.832	61.969	13
$\mathcal{O}_8$	1433	12	108	2.263	1.182	3.445	9.129	12
$\mathcal{O}_9$	2281	16	150	2.954	2.278	5.232	39.772	22
$\mathcal{O}_{10}$	1445	18	80	2.186	1.501	3.687	8.656	25
$\mathcal{O}_{11}$	2218	30	188	2.647	8.11	10.757	239.882	134
$\mathcal{O}_{12}$	7380	31	163	4.513	2.824	7.337	887.284	62
$\mathcal{O}_{13}$	1784	19	280	2.6	1.265	**3.865**	**2.768**	19
$\mathcal{O}_{14}$	11014	12	160	5.322	1.306	6.628	67.826	12
$\mathcal{O}_{15}$	19574	15	660	9.224	3.595	12.819	1803.287	15

[1] ♯*onto*. denotes the number of axioms in an ontology.
[2] ♯*uc*. denotes the number of unsatisfiable classes in an ontology.

5. Conclusions

In this paper, we proposed an optimization method for MinAs computation using the DOBP algorithm. Unlike the DOBP algorithm, our proposed Module-DOBP method first extracts an incoherent-module relevant to the unsatisfiability of the classes, and then implements the DOBP algorithm based on the extracted module. An incoherent-module $\mathcal{M}$ for an incoherent ontology $\mathcal{O}$ and an unsatisfiable class C is a subset of $\mathcal{O}$ that is guaranteed to preserve the unsatisfiability of $\mathcal{O}$. The experimental evaluation showed that the Module-DOBP optimization approach performs better than the DOBP method in most cases. Regarding future work, we will attempt to present other debugging algorithms using the module-based approach.

Acknowledgments

We would like to thank the anonymous referees for their comments. The research presented in this paper was partially supported by the Key Science and Technology Research of Henan Province, China (Grant No. 222102210232, Grant No. 222102210279, Grant No. 212102210516)

Conflict of interest

We declare that we have no financial or personal relationships with other people or organizations that could have inappropriately influenced our work; there is no professional or other personal interest of any nature or kind in any product, service or company that could be construed as having influenced

the position presented in, or the review of, the manuscript entitled, "Effective method for detecting error causes from incoherent biological ontologies".

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.
2. I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, From SHIQ and RDF to OWL: The making of a web ontology language, *J. Web Semantics*, **1** (2003), 7–26. https://doi.org/10.1016/j.websem.2003.07.001
3. J. S. C. Lam, D. Sleeman, J. Z. Pan, W. Vasconcelos, A fine-grained approach to resolving unsatisfiable ontologies, *J. Data Semantics X*, **10** (2008), 62–95. https://doi.org/10.1007/978-3-540-77688-8_3
4. L. Qiu, Y. Liu, Y. Song, B. Zhang, A conflict diagnosis approach of changing sequences in gene ontology evolution, *Int. J. Control Autom.*, **7** (2014), 269–284.
5. X. W. Zhao, X. T. Li, Z. Q. Ma, M. H. Yin, Identify DNA-binding proteins with optimal Chou's amino acid composition, *Proteins Pept. Lett.*, **19** (2012), 398–405. https://doi.org/10.2174/092986612799789404
6. J. Zhang, Y. Zhang, Z. Ma, In silico prediction of human secretory proteins in plasma based on discrete firefly optimization and application to Cancer biomarkers identification, *Front. Genet.*, **10** (2019), 542. https://doi.org/10.3389/fgene.2019.00542
7. J. Zhang, H. Chai, G. Yang, Z. Ma, Prediction of bioluminescent proteins by using sequence-derived features and lineage-specific scheme, *BMC Bioinf.*, **18** (2017), 1–13. https://doi.org/10.1186/s12859-017-1709-6
8. S. Schlobach, Z. Huang, R. Cornet, F. Harmelen, Debugging incoherent terminologies, *J. Autom. Reasoning*, **39** (2007), 317–349. https://doi.org/10.1007/s10817-007-9076-z
9. Y. Zhang, D. Ouyang, Y. Ye, Glass-box debugging algorithm based on unsatisfiable dependent paths, *IEEE Access*, **5** (2017), 18725–18736. https://doi.org/10.1109/ACCESS.2017.2753381
10. F. Simančík, B. Motik, I. Horrocks, Consequence-based and fixed-parameter tractable reasoning in description logics, *Artif. Intell.*, **209** (2014), 29–77. https://doi.org/10.1016/j.artint.2014.01.002
11. Z. Zhou, G. Qi, B. Suntisrivaraporn, A new method of finding all justifications in OWL 2 EL, in *2013 IEEE/WIC/ACM International Conferences on Web Intelligence*, (2013), 213–220. https://doi.org/10.1109/WI-IAT.2013.31
12. X. Fu, G. Qi, Y. Zhang, Z. Zhou, Graph-based approaches to debugging and revision of terminologies in DL-Lite, *Knowl. Based Syst.*, **100** (2016), 1–12. https://doi.org/10.1016/j.knosys.2016.01.039
13. B. C. Grau, I. Horrocks, Y. Kazakov, U. Sattler, A logical framework for modularity of ontologies, in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, (2007), 298–303.

14. B. Grau, I. Horrocks, Y. Kazakov, U. Sattler, Just the right amount: extracting modules from ontologies, in *Proceedings of the 16th international conference on World Wide Web*, (2007), 717–726. https://doi.org/10.1145/1242572.1242669

15. B. Cuenca Grau, C. Halaschek-Wiener, Y. Kazakov, History matters: incremental ontology reasoning using modules, in *Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, (2007), 183–196. https://doi.org/10.1007/978-3-540-76298-0_14

16. A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in *Proceedings of 6th International Semantic Web Conference, ISWC 2007 and 2nd Asian Semantic Web Conference*, (2007), 267–280. https://doi.org/10.1007/978-3-540-76298-0_20

17. M. Horridge, *Justification Based Explanation in Ontologies*, Ph.D thesis, University of Manchester in Manchester, 2011.

18. B. Suntisrivaraporn, Module Extraction and Incremental Classification: A pragmatic approach for $\mathcal{EL}$+ ontologies, in *Proceedings of the 5th European Semantic Web Conference*, (2008), 230–244. https://doi.org/10.1007/978-3-540-68234-9_19

19. J. Du, G. Qi, Q. Ji, Goal-directed module extraction for explaining OWL DL entailments, in *Proceedings of the 8th International Semantic Web Conference*, (2009), 163–179. https://doi.org/10.1007/978-3-642-04930-9_11

20. J. Du, G. Qi, Decomposition-Based Optimization for Debugging of Inconsistent OWL DL Ontologies, in *Proceedings of the 4th International Conference on the Knowledge Science, Engineering and Management*, (2010), 88–100. https://doi.org/10.1007/978-3-642-15280-1_11

21. M. Gao, Y. Ye, D. Ouyang, B. Wang, Finding justifications by approximating core for large-scale ontologies, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI2019)*, (2019), 6432–6433.

22. Y. Zhang, R. Yao, D. Ouyang, J. Gao, F. Liu, Debugging incoherent ontology by extracting a clash module and identifying root unsatisfiable concepts, *Knowl. -Based Syst.*, **223** (2021), 107043. https://doi.org/10.1016/j.knosys.2021.107043

23. Y. Zhang, D. Ouyang, Y. Ye, An optimization strategy for debugging incoherent terminologies in dynamic environments, *IEEE Access*, **5** (2017), 24284–24300. https://doi.org/10.1109/ACCESS.2017.2758521

24. Q. Ji, Z. Gao, Z. Huang, Study of ontology debugging approaches based on the criterion set BLUEI2CI, in *Proceedings of the 6th Chinese Semantic Web Symposium and 1st Chinese Web Science Conference*, (2013), 251–264. https://doi.org/10.1007/978-1-4614-6880-6_22

25. Q. Ji, Z. Gao, Z. Huang, M. Zhu, Measuring effectiveness of ontology debugging systems, *Knowl. -Based Syst.*, **71** (2014), 169–186. https://doi.org/10.1016/j.knosys.2014.07.023

26. Y. Ye, X. Cui, D. Ouyang, Extracting a justification for OWL ontologies by critical axioms, *Front. Comput. Sci.*, **14** (2020), 55–64. https://doi.org/10.1007/s11704-019-7267-5

27. J. Gao, D. Ouyang, Y. Ye, Exploring duality on ontology debugging, *Appl. Intell.*, **50** (2020), 620–633. https://doi.org/10.1007/s10489-019-01528-y

28. J. Du, G. Qi, X. Fu, A practical fine-grained approach to resolving incoherent OWL 2 DL terminologies, in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, (2014), 919–928. https://doi.org/10.1145/2661829.2662046

29. D. Fleischhacker, C. Meilicke, J. Völker, M. Niepert, Computing incoherence explanations for learned ontologies, in *Proceedings of the 7th International Conference on the Web Reasoning and Rule Systems*, (2013), 80–94. https://doi.org/10.1007/978-3-642-39666-3_7

30. G. Flouris, Z. Huang, J. Z. Pan, D. Plexousakis, H. Wache, Inconsistencies, negations and changes in ontologies, in *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, (2006), 1295–1300.

31. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: a practical owl-dl reasoner, *J. Web Semantics*, **5** (2007), 51–53. https://doi.org/10.1016/j.websem.2007.03.004

32. R. Shearer, B. Motik, I. Horrocks, HermiT: a highly-efficient OWL reasoner, in *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008)*, (2008), 1–10.

33. D. Tsarkov, I. Horrocks, FaCT++ description logic reasoner: system description, in *International joint conference on automated reasoning*, (2006), 292–297. https://doi.org/10.1007/11814771_26

34. F. Baader, B. Suntisrivaraporn, Debugging SNOMED CT using axiom pinpointing in the description logic EL, in *Proceedings of the 3rd International Conference on Knowledge Representation in Medicine*, **410** (2008), 1–7.

35. B. C. Grau, I. Horrocks, Y. Kazakov, U. Sattler, Modular reuse of ontologies: theory and practice, *J. Artif. Intell. Res.*, **31** (2008), 273–318. https://doi.org/10.1613/jair.2375

36. H. Wang, M. Horridge, A. Rector, N. Drummond, J. Seidenberg, Debugging OWL-DL ontologies: a heuristic approach, in *Proceedings of the 4th International Semantic Web Conference*, (2005), 745–757. https://doi.org/10.1007/11574620_53

37. Q. Ji, Z. Gao, Z. Huang, M. Zhu, An efficient approach to debugging ontologies based on patterns, in *Proceedings of the Semantic Web-Joint International Semantic Technology Conference*, (2011), 425–433. https://doi.org/10.1007/978-3-642-29923-0_33

38. Ó. Corcho, C. Roussey, L. M. Vilches-Blázquez, I. Perez, Pattern-based OWL ontology debugging guidelines, in *Proceedings of the Workshop on Ontology Patterns (WOP 2009)*, (2009), 1–15.

AIMS Press