IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 29, 2019, accepted May 12, 2019, date of publication May 28, 2019, date of current version June 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919498

Resolving Logical Contradictions in Description Logic Ontologies Based on Integer Linear Programming

QIU JI[1], KHAOULA BOUTOUHAMI[2,3], AND GUILIN QI[2,3]

[1]School of Modern Posts & Institute of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[2]School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
[3]Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 211189, China

Corresponding author: Khaoula Boutouhami (kboutouhami@gmail.com)

This work was supported in part by the National Science Foundation of China under Grant 61602259, in part by the Research Foundation for Advanced Talents of Nanjing University of Posts and Telecommunications under Grant NY216022, in part by the National Key R&D Program of China under Grant 2018YFC0830200, in part by the National Natural Science Foundation of China Key Project under Grant U1736204, in part by the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant 16KJB520033, and in part by the National Engineering Laboratory for Logistics Information Technology, YuanTong Express Co., Ltd.

ABSTRACT When resolving logical contradictions in ontologies, Reiter's hitting set tree algorithm is often applied to satisfy the minimal change principle. To improve the efficiency, the researchers have proposed various algorithms by using a scoring function, defining new semantics or applying some heuristic strategies. However, these algorithms either sacrifice minimal change or are designed for less expressive ontologies like DL-Lite. In this paper, we propose a mathematic approach based on integer linear programming, which is an optimization problem of maximizing or minimizing a linear objective function, to deal with DL ontologies. Specifically, we define the integer linear programming-based model to resolve logical contradictions. To realize the model, we propose one algorithm to find a cardinality-minimal solution and two algorithms dealing with weighted ontologies. Our experiments are conducted over 70 real-life and artificial ontologies to compare our algorithms with those hitting set tree-based ones. Through the experiments, the prominent efficiency and effectiveness have been exhibited by our algorithms. They usually take about 0.4 s to find a solution while others spend more than 100 s in many cases. The experimental results also show that the first two algorithms could find the cardinality-minimal solutions and those with a minimal sum of weights, respectively.

INDEX TERMS Logical contradictions, inconsistency handling, ontology repair, semantic web, integer linear programming.

I. INTRODUCTION

In the Semantic Web, ontologies provide shared and precisely defined terms and play a key role in the formal representation of knowledge [1], [2]. Since building ontologies is an error-prone effort and ontologies often evolve over time, logical contradictions are always unavoidable in many application scenarios like ontology construction, ontology revision and ontology mapping [3]–[6]. Generally, logical contradictions consist of inconsistency and incoherence. An ontology is inconsistent iff it has no model, namely it is inconsistent in the first-order sense. An ontology is incoherent iff there exists some unsatisfiable concept which is interpreted as an empty set. Usually, incoherence occurs in terminologies of ontologies and inconsistency is caused by adding instances of concepts or relations to an incoherent ontology. Since the conclusions derived from an inconsistent ontology using the standard reasoning may be completely meaningless, dealing with logical contradictions becomes an important and meaningful task.

When resolving logical contradictions, it is always desired that the proposed approaches satisfy the minimal change principle. A commonly used approach (e.g., [7]–[9]) to reach this goal is utilizing Reiter's Hitting Set Tree (HST)

The associate editor coordinating the review of this manuscript and approving it for publication was Wajahat Ali Khan.

algorithm [10]. It first computes a set of or all of the minimal sets of axioms that preserve the incoherence or inconsistency and then construct a HST to find a minimal set of axioms for removing. Since it is very expensive to expand all branches in a HST, the researchers provide algorithms to reduce the search space by using scoring function or considering weights (e.g., [5]). Other algorithms define new semantics (e.g., [11]) to avoid computing minimal sets of axioms, or applying some heuristic strategies to find a solution that may not be minimal (e.g., [12], [13]). However, these algorithms either sacrifice the minimal change principle or are designed for less expressive ontologies like DL-Lite.

In this paper, we propose a mathematic approach based on integer linear programming (ILP), which is an optimization problem of maximizing or minimizing a linear objective function. The ILP-based models have been used for many tasks in the Semantic Web such as fuzzy ontology reasoning (e.g. [14], [15]), ontology matching (e.g. [16], [17]) and knowledge base embeddings (e.g. [18]), because it is very effective in solving problems with complex constraints. To be specific, we first define the ILP-based model to resolve logical contradictions. To realize the model, we then propose one algorithm to find a cardinality-minimal solution and two algorithms dealing with weighted ontologies. To see the performance, a comprehensive evaluation has been conducted over 70 ontologies which are real-life ontologies or artificial ones constructed by our generator. We compare our algorithms with those HST-based ones w.r.t. the efficiency and effectiveness. Through the experimental results, it is revealed that our algorithms have shown their prominent efficiency and effectiveness. They often took around 0.4 seconds to find a solution while others spent more than 100 seconds in many cases. The results also show that the first two algorithms could find the cardinality-minimal solutions and those with minimal sum of weights respectively. Besides, all found solutions have been verified to be correct.

In summary, our paper contains the following contributions:

1) We provide a general model to resolve logical contradictions based on integer linear programming.
2) We propose one algorithm to find a cardinality-minimal solution and two algorithms to keep the information of weights as much as possible.
3) An extensive evaluation is conducted over 70 ontologies to see the performance of our algorithms w.r.t. the effectiveness and efficiency.

The rest of this paper is organized as follows: Section II provides a preliminary introduction and Section III presents our proposed approach by introducing the model and three algorithms. The experiment preparing and evaluation results are reported in Section IV and Section V separately. Finally, we conclude the paper and provide future work in Section VII.

II. PRELIMINARIES

In this section, we provide a brief introduction to Description Logics (DLs) and the key notions of logical contradictions and integer linear programming. More details can be found in [19]–[21].

A. DESCRIPTION LOGICS

In DLs, there are three kinds of entities: concepts representing sets of individuals, roles representing binary relations between the individuals and individual names representing single individuals in the domain. A DL ontology $O = (\mathcal{T}, \mathcal{A})$ consists of a finite set $\mathcal{T}$ (TBox) of terminology axioms and a set $\mathcal{A}$ (ABox) of individual axioms. A TBox includes concept axioms and role axioms. Concept axioms have the form $C \sqsubseteq D$ where C and D are (possibly complex) concept descriptions.[1] Role axioms are expressions of the form $R \sqsubseteq S$, where R and S are role descriptions. An ABox contains concept assertions of the form $C(a)$ and role assertions of the form $R(a, b)$, where C is a concept, R is a role, a and b are individual names.

The semantics of DLs is defined via a model-theoretic semantics, which explicates the relationship between the language syntax and the model of a domain: An interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain set $\triangle^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which maps from concepts and roles to subsets of the domain and binary relations on the domain respectively. An interpretation $\mathcal{I}$ satisfies a concept axiom $C \subseteq D$ (a concept assertion $C(a)$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($a^{\mathcal{I}} \in C^{\mathcal{I}}$, respectively). It satisfies a role axiom $R \sqsubseteq S$ (a role assertion $R(a, b)$) if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, respectively). An interpretation $\mathcal{I}$ is called a *model* of an ontology, iff it satisfies each axiom in the ontology. A named concept C in an ontology O is unsatisfiable iff, for each model $\mathcal{I}$ of O, $C^{\mathcal{I}} = \emptyset$. An ontology is incoherent, iff there exists an unsatisfiable named concept in it. An ontology is inconsistent iff it has no model. We use the general concept of logical contradictions to indicate the unsatisfiability, incoherence or inconsistency.

B. LOGICAL CONTRADICTIONS IN DESCRIPTION LOGICS

Usually, minimal sets of axioms for explaining logical contradictions are often required to be computed first and then a solution of axioms needs to be found such that removing these axioms can resolve the contradictions. In the following, we provide the key notations that are useful to explain or resolve logical contradictions.

Definition 1 ((MUPS) [22]): Let C be an unsatisfiable concept in an ontology O. An ontology $O' \subseteq O$ is a *minimal unsatisfiability-preserving sub-ontology (MUPS)* of O w.r.t. C if C is unsatisfiable in O' and satisfiable in every sub-ontology $O'' \subset O'$.

A MUPS of O *w.r.t.* C is a minimal sub-ontology of O in which C is unsatisfiable. MUPSs are useful for relating

[1] A complex concept is a concept that is formed by some atomic concepts and constructors such as conjunction $\sqcap$ and disjunction $\sqcup$.

sets of axioms to the unsatisfiability of specific concepts. For relating sets of axioms to the incoherence of an ontology in general, a minimal incoherence-preserving sub-ontology is defined.

Definition 2 ((MIPS) [22]): Let O be an incoherent ontology. An ontology $O' \subseteq O$ is a *minimal incoherence-preserving sub-ontology (MIPS)* of O if O' is incoherent and every sub-ontology $O'' \subset O'$ is coherent.

A MIPS of O is a minimal incoherent sub-ontology of O. Note that a MIPS must be a MUPS, but not vice versa.

Definition 3 ((MIS) [23]): An ontology $O' \subseteq O$ is a minimal inconsistent sub-ontology (MIS) of O, if O' is inconsistent and every sub-ontology $O'' \subset O'$ is consistent.

To conclude, a logical contradiction of an ontology can be a MUPS, MIPS or MIS. Practical algorithms to calculate MUPS, MIPS and MIS of a given ontology can be found in [24]. For convenience, we use a conflict to indicate a MUPS, MIPS or MIS. In general, resolving the given logical contradictions in an ontology is to compute a solution based on the corresponding conflicts such that removing the axioms in the solution could break each conflict. Breaking a conflict means removing at least one axiom from the conflict. Such a solution is also called a diagnosis [6].

C. INTEGER LINEAR PROGRAMMING

Linear programming is used to obtain the most optimal solution for a problem with given constraints. A linear program is an optimization problem of maximizing or minimizing a linear objective function of variables that are subject to a set of constraints expressed as linear equations or inequations. When an optimization problem can be modeled as a linear program whose variables are forced to be integers, then the program is called integer linear programming (ILP).[2]

In this paper, we use the pure 0-1 ILP. Given n boolean variables and m linear constraints, the problem of 0-1 ILP is to find an assignment of either 0 or 1 to the variables such that all constraints are satisfied. A 0-1 ILP optimization problem can be stated in the following general form:

$$Minimize/Maximize\ z = \mathbf{c}\mathbf{x}^T = \sum_{j=1}^{n} w_j x_j \tag{1a}$$

$$subject\ to\ A\mathbf{x}^T \geqslant \mathbf{b}^T \tag{1b}$$

$$and\ x_j \in \{0, 1\}\ j = 1, \ldots, n \tag{1c}$$

Here, $\mathbf{x} = (x_1, \ldots, x_n)$ represents the set of decision variables to be determined and each element x_j can have value 0 or 1. $\mathbf{c} = (w_1, \ldots, w_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ are vectors of coefficients and each element of $\mathbf{b}$ is an integer. A is a $m \times n$ matrix of coefficients. The values for the coefficients will be determined by specific application scenarios.

The objective function (1a) describes a criterion (or a measure) that we wish to minimize (e.g., cost) or maximize (e.g., profit). The limitations that restrict our choices for decision variables are described using mathematical constraints (1b). In many practical problems, people's decisions are often restricted by the limitations like resource limitation and physical, strategic or economical constraints. It is noted that, the complexity of solving an ILP problem is NP-complete and it depends on the number of variables [25].

[2]https://en.wikipedia.org/wiki/Integer_linear_programming

III. THE ILP-BASED APPROACH

In this section, we define the model of resolving logical contradictions and also provide specific algorithms.

A. MODEL FOR RESOLVING LOGICAL CONTRADICTIONS

To satisfy the minimal change principle, we define an incision function below. This is inspired by the incision function given in [5] to revise an ontology.

Definition 4 ((Incision Function)): Let O be an ontology and $CONF(O)$ be a set of conflicts in O. An incision function σ for O is a function $2^{2^O} \rightarrow 2^O$ such that:

(i) $\sigma(CONF(O)) \subseteq \bigcup_{conf \in CONF(O)} conf$;
(ii) if $conf \in CONF(O)$, then $conf \cap \sigma(CONF(O)) \neq \emptyset$.

The incision function for O selects a set of axioms to break every conflict. The first condition ensures that the selected axioms (i.e. axioms in $\sigma(CONF(O))$) belong to the union of all of the given conflicts. Namely, we do not remove any axiom that does not belong to any conflict. The second condition shows that $\sigma(CONF(O))$ must contain at least one axiom from each conflict. This ensures every conflict could be broken. With the two conditions, once the axioms in $\sigma(CONF(O))$ are removed from O, all contradictions in the given set $CONF(O)$ will not exist any more.

Similar to the work in [5], we define minimal incision function to select the minimal set of axioms for removing.

Definition 5 (Minimal Incision Function): Let O be an ontology and $CONF(O)$ be a set of conflicts in O. An incision function σ for O is minimal if there is no other incision function σ' for O such that $\sigma'(CONF(O)) \subset \sigma(CONF(O))$.

The minimal incision functions may contain different number of axioms. To make the number of selected axioms minimal, a cardinality-minimal incision function is defined.

Definition 6 (Cardinality-Minimal Incision Function): Let O be an ontology and $CONF(O)$ be a set of conflicts in O. An incision function σ for O is cardinality-minimal if there is no other incision function σ' for O such that $|\sigma'(CONF(O))| < |\sigma(CONF(O))|$.

Clearly, a cardinality-minimal incision function is always a minimal incision function.

To find a specific incision function, we apply 0-1 ILP by the following definition.

Definition 7 (ILP Model): Let $CONF(O) = \{conf_1, conf_2, \ldots, conf_m\}$ be a set of conflicts in an ontology O and $S_{union} = \{ax_1, ax_2, \ldots, ax_n\}$ include all distinct axioms in the conflicts. For each axiom ax_j $(j = 1, \ldots, n)$ in S_{union}, a binary variable x_j is associated, whose value can be 0 or 1. The problem of resolving logical contradictions in $CONF(O)$ is to find an assignment to all variables which can be modeled

Algorithm 1: The ILP-Based Algorithm - Flat Case

Data: An ontology O and a set of conflicts $CONF(O)$
Result: A solution of axioms

1 **begin**
2 $C := \emptyset$
3 $S_{union} := \bigcup_{conf \in CONF(O)} conf$
4 $X := \{x_j | ax_j \in S_{union},\ j = 1, \ldots, |S_{union}|\}$
5 $z := \sum_{x_j \in X} x_j$
6 **for** $conf \in CONF(O)$ **do**
7 $X_{conf} := \{x_j | ax_j \in conf, x_j \in X\}$
8 $c_i := (\sum_{x_j \in X_{conf}} x_j) \geq 1$
9 $C := C \cup \{c_i\}$
10 $S_{assi} := ILP\_Solver(z, C, \min)$
11 $\sigma_1(CONF(O)) := \{ax_j | (x_j = 1) \in S_{assi}\}$
12 **return** $\sigma_1(CONF(O))$
13 **end**

by the following ILP.

$$Minimize\ z = \sum_{j=1}^{n} w_j x_j \tag{2a}$$

$$subject\ to\ c_i : \sum_{j=1}^{n} a_{ij} x_j \geqslant 1, \quad i = 1, \ldots, m. \tag{2b}$$

Here, w_j and a_{ij} are coefficients to be determined by specific algorithms given in next section. Normally, a_{ij} has value 1 if $ax_j \in conf_i$, and 0 otherwise. If an axiom in a conflict is not removable for some reason, its corresponding variable will be set to be 0 directly.

In Definition 7, each conflict corresponds to a constraint which ensures at least one axiom should be removed from each conflict. Once all coefficients have been determined, a traditional ILP solver could be applied to find an assignment to all variables under the given constraints. In this work, we use the commercial linear programming tool Cplex which is an optimization software developed by IBM ILOG.[3]

B. ALGORITHMS FOR RESOLVING LOGICAL CONTRADICTIONS

Since the coefficients in Definition 7 have not been determined, no specific incision function can be defined up to now. In the following, we propose three algorithms to resolve logical contradictions by computing specific incision functions.

1) ILP-BASED ALGORITHM FOR RESOLVING LOGICAL CONTRADICTIONS - FLAT CASE

The first algorithm (see Algorithm 1) takes an ontology and a set of conflicts as inputs and outputs a solution with selected axioms for removing.

In the algorithm, all distinct axioms in the conflicts are obtained each of which is associated with a binary variable (see lines 3 and 4). To construct the objective function, all axioms are regarded as the same important and thus 1 is assigned to the coefficients in the function (see Line 5). After that, all constraints can be constructed by assuming all axioms are removable (see lines from 6 to 9). Namely, once an axiom ax_j is contained in a conflict $conf_i$, its variable x_j should appear in the corresponding constraint. Afterwards, the function $ILP\_Solver(z, C, \min)$ is invoked to generate an optimal assignment (see Line 10), where the parameter *min* indicates minimizing the objective function. The returned assignment S_{assi} is the first one found by the ILP solver. Based on this assignment, we can find those axioms whose corresponding variables have value 1 to form a final solution $\sigma_1(CONF(O))$ (see Line 11). Note that, if the set of conflicts includes all MIPS or MIS in O, then removing the axioms in the final solution makes O coherent or consistent respectively.

An example is given below for better understanding Algorithm 1.

Example 1: Suppose we have an ontology $O = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$ and the set of all MIPS in O: $CONF(O) = \{conf_1, conf_2, conf_3, conf_4, conf_5\}$, where $conf_1 = \{a_1, a_2, a_3\}$, $conf_2 = \{a_1, a_2, a_4\}$, $conf_3 = \{a_1, a_7\}$, $conf_4 = \{a_2, a_5, a_6\}$, $conf_5 = \{a_7, a_8\}$. We can obtain S_{union}= $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$ and associate a variable to each axiom in the set. The objective function and constraints can be constructed as below:

$$z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$$

$$c_1 : x_1 + x_2 + x_3 \geq 1$$

$$c_2 : x_1 + x_2 + x_4 \geq 1$$

$$c_3 : x_1 + x_7 \geq 1$$

$$c_4 : x_2 + x_5 + x_6 \geq 1$$

$$c_5 : x_7 + x_8 \geq 1$$

By applying $ILP\_Solver(z, C, \min)$, we can find an assignment consisting of $x_2 = 1$, $x_7 = 1$ and 0 for other variables and then obtain the final solution $\sigma_1(CONF(O)) = \{a_2, a_7\}$. We can see that a_2 breaks three conflicts (i.e., $conf_1$, $conf_2$ and $conf_4$) and a_7 breaks the two remaining conflicts. Namely, removing the axioms in the solution could resolve all given contradictions.

Proposition 1: Algorithm 1 computes a cardinality-minimal incision function σ_1.

Proof: We first prove σ_1 is an incision function according to the two conditions given in Definition 4.

(i) According to Algorithm 1, we know that S_{assi} is an assignment of 0 or 1 to all variables in X and each variable has a corresponding axiom in S_{union}. Clearly, $\{ax_j | (x_j = 1) \in S_{assi}\} \subseteq S_{union}$. Namely, $\sigma_1(CONF(O)) \subseteq \bigcup_{conf \in CONF(O)} conf$.

(ii) For a conflict $conf \in CONF(O)$, Algorithm 1 constructs a constraint $(\sum_{x_j \in X_{conf}} x_j) \geq 1$ (see Line 8) which should be satisfied by the found assignment S_{assi}. It means that at least one variable in X_{conf} has been assigned to

[3]https://www.ibm.com/analytics/cplex-optimizer

Algorithm 2: The ILP-Based Algorithm - Weighted Case

Data: A weighted ontology O and a set of conflicts $CONF(O)$
Result: A solution of axioms

1 **begin**
2 $C := \emptyset$
3 $S_{union} := \bigcup_{conf \in CONF(O)} conf$
4 $X := \{x_j | ax_j \in S_{union},\ j = 1, \ldots, |S_{union}|\}$
5 $z := \sum_{x_j \in X} w_j.x_j$
6 **for** $conf \in CONF(O)$ **do**
7 $X_{conf} := \{x_j | ax_j \in conf, x_j \in X\}$
8 $c_i := (\sum_{x_j \in X_{conf}} x_j) \geq 1$
9 $C := C \cup \{c_i\}$
10 $S_{assi} := ILP\_Solver(z, C, \min)$
11 $\sigma_2(CONF(O)) := \{ax_j | (x_j = 1) \in S_{assi}\}$
12 **return** $\sigma_2(CONF(O))$
13 **end**

be 1 according to S_{assi}. Thus, at least one axiom in $conf$ should be included in $\{ax_j | (x_j = 1) \in S_{assi}\}$). Namely, $conf \cap \sigma_1(CONF(O)) \neq \emptyset$.

Second, we prove σ_1 is cardinality-minimal. In Algorithm 1, the objective function actually computes the number of variables with value 1. As each variable corresponds to an axiom in S_{union}, the optimization purpose of minimizing z under the given constraints is to find a minimal number of axioms for breaking all given conflicts. It also means to find a cardinality-minimal solution $\sigma_1(CONF(O))$. Therefore, σ_1 is a cardinality-minimal incision function.

□

2) ILP-BASED ALGORITHM FOR RESOLVING LOGICAL CONTRADICTIONS - WEIGHTED CASE

To deal with the ontologies with weights, we intend to resolve logical contradictions by deleting axioms as minimal as possible and keeping information of weights as much as possible. Assume each axiom ax_j has a weight $w_j \in (0, 1]$. A straightforward way to make use of weights in an ILP program is to modify the objective function in Algorithm 1 to the following function:

$$z := \sum_{x_i \in X} w_i.x_i$$

Algorithm 2 presents the weighted case, where the weights can be used as the coefficients in the objective function (see Line 5). Since all axioms are also considered as removable, the constraints are constructed in a similar way as Algorithm 1. It is noted that, as a variable can only have value 0 or 1, the optimization objective is actually minimizing the sum of weights for those axioms in a final solution.

Proposition 2: Algorithm 2 computes a minimal incision function σ_2.

Proof: We can prove σ_2 is an incision function in a similar way as Proposition 1. Now, we prove σ_2 is a minimal incision function. Suppose we can find another incision function σ' to resolve the contradictions in $CONF(O)$ such that $\sigma'(CONF(O)) \subset \sigma_2(CONF(O))$. We then obtain $\sum_{ax_i \in \sigma'(CONF(O))} w_i < \sum_{ax_i \in \sigma_2(CONF(O))} w_i$. This contradicts with the optimization objective which is to minimize $\sum_{x_i \in X} w_i.x_i$ (namely minimize the sum of weights for those axioms in the final solution). Therefore, σ' does not exist and σ_2 is a minimal incision function.

□

3) ILP-BASED ALGORITHM FOR RESOLVING LOGICAL CONTRADICTIONS-WEIGHTED CASE WITH THRESHOLD

In many application cases like ontology mapping repair [16] and uncertain reasoning in possibilistic logic [21], it is desired to keep those axioms with more important or higher weights by using a threshold. Namely, only the axioms with the weights that are no more than the threshold could be removed. Thus, we propose Algorithm 3 to take the threshold into consideration.

In this algorithm, the objective function is constructed in the same way as Algorithm 2 (see Line 6). Before constructing the constraints, we need to know which axioms are removable and obtain the corresponding variables $X_{removable}$ (see the lines from 7 to 9). As we would like to keep those axioms with weights above a threshold, all other axioms are regarded as removable. In such cases, we will fail to find a solution to resolve all given contradictions if there exists a conflict $conf$ such that X_{conf} has no overlapping with $X_{removable}$. It means all axioms in $conf$ are not removable and this conflict cannot be broken. If this happens, an empty set will be returned (see lines from 11 to 13). Otherwise, a constraint could be successfully constructed for the removable axioms in a conflict (see Line 14). Namely, all other variables have been determined to have value 0.

In order to avoid the case of failing to find any solution by Algorithm 3, choosing an appropriate threshold is critical. The assignment of this value must ensure that the minimum value of the weights associated to the axioms in each conflict should be no more than α. In this way, we can ensure that at least one axiom in each conflict is removable. Such a threshold can be formally defined as below.

Definition 8 (Effective Threshold): Given an ontology O and a set of conflicts $CONF(O)$, S_{union} is used to indicate the union of all given conflicts. An effective threshold α for resolving the given conflicts should satisfy the following condition for any $conf_i \in CONF(O)$:

$$\alpha \geq \min(\{w_j \mid ax_j \in conf_i\}).$$

Definition 8 means a threshold is effective if it is no less than each minimal weight in a conflict. In another word, if the threshold is less than the minimal weight in a conflict, it means that all axioms in this conflict cannot be deleted and we fail to resolve this conflict.

Algorithm 3: The ILP-Based Algorithm - Weighted Case With Threshold

Data: A weighted ontology O, a set of conflicts $CONF(O)$ and a threshold α
Result: A solution of axioms

```
begin
    C := ∅
    X_removable := ∅
    S_union := ⋃_{conf∈CONF(O)} conf
    X_all := {x_j | ax_j ∈ S_union, j = 1, ..., |S_union|}
    z := Σ_{x_j∈X_all} w_j.x_j
    for ax_j ∈ S_union do
        if w_j ≤ α then
            X_removable := X_removable ∪ {x_j}
    for conf ∈ CONF(O) do
        X_conf := {x_j | ax_j ∈ conf, x_j ∈ X_all}
        if X_conf ⋂ X_removable = ∅ then
            return ∅
        c_i := (Σ_{x_j∈(X_conf ⋂ X_removable)} x_j) ≥ 1
        C := C ∪ {c_i}
    S_assi := ILP_Solver(z, C, min)
    σ3(CONF(O)) := {ax_j | (x_j = 1) ∈ S_assi}
    return σ3(CONF(O))
end
```

To find such a threshold, we could use the minmax measure to find its lowest bound:

$$\alpha = \max_{c \in CONF(O)} (\{w \mid w = \min(\{w_j \mid ax_j \in c\})\})$$

Namely, α is the maximal value among all weights that are minimal in a conflict.

Example 2: Let us consider again the ontology from Example 1. Suppose a weight is attached to each axiom in the conflicts as follows:

$$w_1 = 0.11, \quad w_2 = 0.55, \; w_3 = 0.19, \; w_4 = 0.77,$$

$$w_5 = 0.30, \quad w_6 = 0.80, \; w_7 = 0.95, \; w_8 = 0.68.$$

Then we can determine the lowest bound of α by applying the minmax measure:

$$\begin{aligned} \alpha &= \max(\{\min(\{0.11, 0.55, 0.19\}), \min(\{0.11.0.95\}), \\ &\quad \min(\{0.11, 0.55, 0.77\}), \min(\{0.55, 0.30, 0.80\}), \\ &\quad \min(\{0.95, 0.68\})\}) \\ &= \max(\{0.11, 0.30, 0.68\}) \\ &= 0.68 \end{aligned}$$

Proposition 3: Algorithm 3 computes an incision function σ_3 if the input α is an effective threshold.

Proof: According to the definition of an incision function, we prove that σ_3 satisfies the two conditions in the following.

(i) Since S_{assi} is an assignment to assign 0 or 1 to all variables in X_{all} and each variable has a corresponding axiom in S_{union}, it is clear that $\{ax_j | (x_j = 1) \in S_{assi}\} \subseteq S_{union}$. Namely, $\sigma_3(CONF(O)) \subseteq \bigcup_{conf \in CONF(O)} conf$.

(ii) As an effective threshold is no less than the minimal weight in any conflict, it means at least one axiom with the minimal weight in a conflict is removable and the corresponding variable should be contained in $X_{removable}$. Namely, for $conf \in CONF(O)$, we have $X_{conf} \bigcap X_{removable} \neq \emptyset$. Since the constraint $(\sum_{x_j \in (X_{conf} \bigcap X_{removable})} x_j) \geq 1$ must be satisfied, at least one variable in $X_{conf} \bigcap X_{removable}$ could be associated with value 1 by the solver. Thus, at least one axiom in $conf$ will be included in $\sigma_3(CONF(O))$ and we have $conf \cap \sigma_3(CONF(O)) \neq \emptyset$. □

IV. EXPERIMENT PREPARING

This section introduces the experimental settings, real-life data sets and artificial data sets respectively. All proposed algorithms have been implemented with OWL API[4] which is widely used in ontology-based applications. The implementation, data sets and the experimental results can be downloaded from a website.[5]

A. EXPERIMENTAL SETTINGS

Our evaluation is performed on a laptop with 2.4 GHz Intel(R) Core(TM)2 Duo CPU and 8.0 GB of RAM using 64 bit operating system Windows 7. The maximum heap space is set to 4 GB and a time limit of 1000 seconds is set to compute MUPS for an unsatisfiable concept[6] or compute a solution to resolve the given contradictions for practical consideration. We use the relevance-based ontology debugging algorithm given in [26] to compute MUPS.

To better see the performance of our algorithms, we compare them with the traditional ones using the hitting set tree (HST) algorithm to satisfy the minimal change principle. For simplicity, the following abbreviations are used to indicate all algorithms mentioned in this section.

- **Alg1:** Algorithm 1 - flat case.
- **Alg2:** Algorithm 2 - weighted case.
- **Alg3:** Algorithm 3 - weighted case with threshold.
- **Hst:** HST algorithm.
- **HstScore:** HST algorithm plus scoring function [5].
- **HstSwoop:** HST algorithm considering weights [8].
- **HstWeight:** HST algorithm plus weights [5].

Hst applies the HST algorithm directly on all given conflicts. HstSwoop finds a solution with minimal sum of weights by modifying the HST algorithm to use the weights of axioms. It is implemented in SWOOP [27] and we update it for being compatible with the new version of OWL API. HstScore applies the HST algorithm to the sets of axioms

[4] http://owlapi.sourceforge.net/

[5] https://github.com/qiuji123/ilpConflicts

[6] Although our approach can be applied to deal with incoherence or inconsistency, we only focus on the incoherence since it is similar to deal with the latter by changing the input conflicts.

each of which contains the axioms with the highest score[7] in a conflict. HstWeight applies the HST algorithm to the sets each of which contains the axioms with the lowest weight in a conflict.

B. REAL-LIFE DATA SETS

In order to test Alg1, we select 6 real-life incoherent ontologies containing relatively more MUPS from the data set given in [24]. Among them, ontologies CHEM-A and miniTambis, marked as O_{r2} and O_{r6} separately, are often used for ontology debugging or repair tasks. Ontology km1500 (marked as O_{r3}) was originally generated by using the ontology learning algorithms. This ontology contains 1375 unsatisfiable concepts many of which have more than 200 MUPS. Other ontologies were constructed by merging two source ontologies and a mapping between them, namely ontologies AROMA-cmt-cocus, LogMapLt-cocus-crs_dr and MaasMatch-cmt-sigkdd (marked as O_{r1}, O_{r4} and O_{r5} respectively) containing highly overlapping MUPS. Such an ontology has quite a lot unsatisfiable concepts and each concept may have more than 100 MUPS.

To test our algorithms considering weights, we use the real-life ontologies provided by OAEI (Ontology Alignment Evaluation Initiative) in 2018 since the axioms in a mapping are often associated with weights. OAEI is a well-known activity in ontology matching area to provide a platform for comparing various ontology matching systems and we use the ontologies provided by the conference track.[8] In this track, there are 16 source ontologies and 12 systems to provide matching results. We merge a mapping with the corresponding source ontologies by translating it to ontology axioms. We then keep those consistent but incoherent merged ontologies with mappings generated by the systems KEPLER, Lily, SANOM and XMap due to different weights associated to the correspondences in a generated mapping. These merged ontologies have similar structure of MUPS and thus we randomly select 9 ontologies. As the source ontoloiges are always regarded as more reliable than their mappings, the weight 1.0 is associated to the axioms in the source ontoloiges. Additionally, we associate a random weight in $(0, 1]$ to each axiom in O_{r3} (i.e., km1500-5000) for testing the scalability of the compared algorithms.

C. ARTIFICIAL DATA SETS

We also provide artificial incoherent ontologies to exhibit various relations among MUPS. Figure 1 presents four typical solution patterns with different relations of MUPS. For convenience, the four patterns are marked as Pattern1, Pattern2, Pattern3 and Pattern4 separately. We explain how to generate ontologies in the following while the specific information about the generated ontologies can be found in Section V. In Figure 1, the circles indicate logical axioms and a rectangle presents a MUPS. The circles with numbers indicate those axioms that may appear in a solution satisfying the minimal change principle. Those circles with the same number indicate the same axiom and other circles represent mutually different ones. Since our algorithms are independent on the MUPS patterns, we use a basic and commonly used pattern to generate MUPS. Namely, a concept is unsatisfiable if it is a subclass of two disjoint concepts [28].

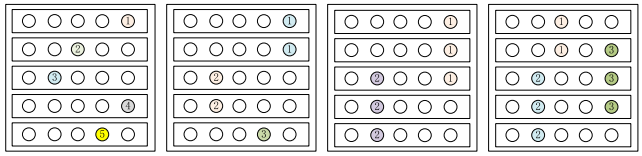

FIGURE 1. The patterns of solutions in the incoherent ontologies constructed by our generator.

Specifically, Pattern1 can be used to construct isolated MUPS, namely no axioms are shared by any two MUPS in an ontology. Our generator starts with an empty ontology and creates new concepts and axioms when constructing a new MUPS. Other patterns are used to generate overlapping MUPS with a desired number of axioms in a cardinality-minimal solution. For example, if a user prefers to generate a set of MUPS containing a cardinality-minimal solution S, then $|S|$ is the desired number of axioms. For convenience, we use cardMin to indicate such a number.

We generate the overlapping MUPS with Pattern2, Pattern3 or Pattern4 in a similar way. Namely, to generate m overlapping MUPS with cardMin $= n$, we need to construct n groups of MUPS such that the number of MUPS in all groups equals to m and the MUPS in the same group shares one axiom. It is noted that, the MUPS in any two groups generated by Pattern2 have no overlapping. The only difference between Pattern2 and Pattern3 is that Pattern3 requires that there is one MUPS containing the shared axioms from two groups. For Pattern4, there are two special groups of MUPS (marked as G_1 and G_2). If G_1 contains more MUPS than G_2, it is also required to have another axiom shared by $|G_1| - 1$ MUPS in G_1 and one MUPS in G_2. In this way, there exists a MUPS containing one axiom with the highest score and another axiom that should be included in a cardinality-minimal solution.

V. EXPERIMENTAL RESULTS

In this section, we provide an exhaustive evaluation of our proposed algorithms with respect to efficiency and effectiveness. The effectiveness measures the number of removed axioms and the correctness of solutions. Specifically, we first compare the algorithms without considering weights based on artificial ontologies with isolated MUPS and then ontologies with overlapping MUPS. Afterwards, we compare other algorithms based on the real-life ontologies with weights.

[7] A score of an axiom means the frequency that the axiom appears in all given conflicts.

[8] http://oaei.ontologymatching.org/2018/results/conference/index.html

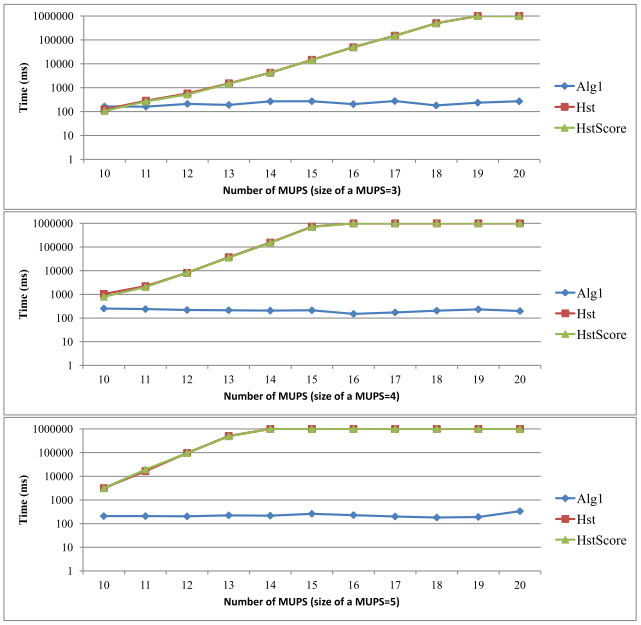

FIGURE 2. The time to find one solution for different number of MUPS with pattern1.

A. RESULTS OF ONTOLOGIES WITH ISOLATED MUPS

This experiment is mainly used to see the performance of the algorithms without using weights, i.e., **Alg1**, **Hst** and **HstScore**, by varying the number of MUPS and the size of a MUPS. Figure 2 illustrates the experimental results based on the ontologies generated with **Pattern1**. In each sub-figure, all MUPS in the ontologies have the same number of axioms. Namely, there are 3, 4 and 5 axioms in a MUPS for the ontologies in the three sub-figures respectively.

From the figure we can observe that, **Alg1** obviously outperforms other algorithms. No matter how the number of MUPS varies or the size of a MUPS changes, the efficiency of our algorithm keeps stable. It often spent no more than 0.4 seconds to find a solution for the given logical contradictions. As for **Hst** and **HstScore**, they have similar performance as expected. It is because the test ontologies only contain isolated MUPS (i.e., no axioms are shared by any two MUPS) and all axioms in the MUPS of an ontology have the same scores that equal to the number of MUPS. Thus, the HST algorithm will be applied to the same sets of axioms and both algorithms perform similarly. Comparing the three sub-figures in Figure 2, **Hst** and **HstScore** cannot deal with the ontologies with more than 18, 14 and 13 isolated MUPS when the size of a MUPS is 3, 4 and 5 separately. It shows that the efficiency of the two algorithms is largely influenced by both the number of MUPS and the size of a MUPS. They spent more than 100 seconds in many cases to find a solution.

As for the effectiveness, these algorithms remove the same number of axioms because the MUPS in an ontology do not share any axioms. Take the ontology with 13 isolated MUPS and 4 axioms in a MUPS as an example. Each of the algorithms removes 13 axioms for breaking the given

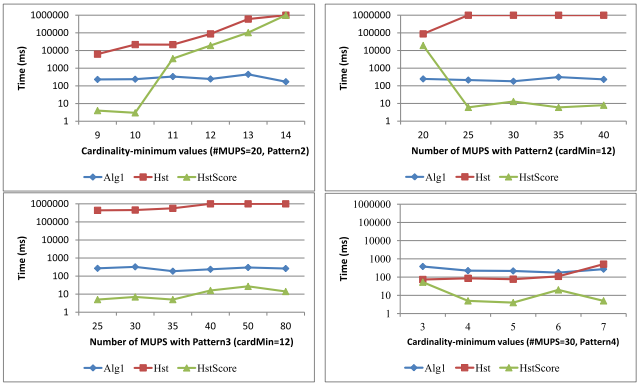

FIGURE 3. The time to find a solution for ontologies with overlapping MUPS.

MUPS. To check the correctness of a solution for a set of conflicts, we need to know whether there is an overlapping between a solution and each conflict or not. Once a conflict does not contain any axiom in the solution, the conflict cannot be broken and the solution fails to resolve all given contradictions. In this way, all found solutions have been verified to be correct.

B. RESULTS OF ONTOLOGIES WITH OVERLAPPING MUPS

To better observe the difference of the algorithms to be compared, this experiment is performed on overlapping MUPS generated by different solution patterns (see Figure 3), where each MUPS contains 5 axioms.

From the figure we can see that, the efficiency of **Alg1** is still very high and stable. That is, each solution was found within 0.5 seconds which were mainly used to create the ILP-based model. We can also see that **Hst** and **HstScore** may handle those MUPS when **cardMin** is less than 14 in many cases. For those ontologies that both algorithms could deal with, **HstScore** is much more efficient. **HstScore** often took no more than 0.1 seconds to finish the process while **Hst** spent more than 10 seconds. This can be explained by the sets of axioms where the HST algorithm is applied to. Since the MUPS in an ontology in Figure 3 have overlapping, the search space for the HST algorithm in **HstScore** has been largely reduced. In overall, the efficiency of both HST-based algorithms is largely influenced by the minimal depth of the branches. For **Hst**, the number of MUPS and the size of a MUPS are also main reasons to influence its efficiency.

As for the effectiveness, **Alg1** is able to find a cardinality-minimal solution for an ontology as expected. **Hst** could also compute such kind of solutions since it traverses all branches of a HST and all cardinality-minimal solutions are included. For **HstScore**, it can find a cardinality-minimal solution for an ontology constructed with all patterns except **Pattern4**. Namely, for an ontology constructed with **Pattern4**, **HstScore** always removes one more axiom than other two algorithms. This can be explained by the solution pattern itself (see **Pattern4** in Figure 1). In this pattern, removing the

TABLE 1. The experimental results for real-life ontologies.

O	$\|O\|$	UC All/Sel.	MUPS Num	# Removed Axioms Alg1	Hst	HstScore	Time (ms) Alg1	Hst	HstScore
O_{r1}	535	62/6	382	4	4	6	287	339	255
O_{r2}	114	37/37	412	1	1	1	299	23	116
O_{r3}	5000	1375/10	1620	8	–	18	460	TO	2010
O_{r4}	234	51/6	225	2	2	8	431	65	80
O_{r5}	379	55/6	309	3	3	5	335	124	102
O_{r6}	173	30/30	28	3	3	3	275	18	22

axioms with number 1 and 2 is enough to resolve the given contradictions. Such a cardinality-minimal solution can be found by Alg1 and Hst. However, HstScore removes one more axiom, namely the axiom with number 3, because its score is the highest in the second MUPS.

Table 1 presents the experimental results for real-life ontologies, where "TO" means time out and "-" indicates unknown. The number of MUPS in the fourth column indicates the number of all distinct MUPS found for the selected unsatisfiable concepts within limited time. According to this table, similar conclusions could be obtained as those for our artificial data sets. It should be noted that, highly overlapping MUPS are contained in these ontologies. The result of O_{r3} proves again that the minimal depth of the branches in a HST is the main factor to influence the efficiency of HstScore, not the number of MUPS. For this ontology, HstScore only spent about 2 seconds to find a solution while Hst cannot finish the process within the limited time (i.e., 1000 seconds).

C. RESULTS OF ONTOLOGIES WITH WEIGHTS

This experiment is used to see the performance of four algorithms considering weights, i.e., Alg2, Alg3, HstSwoop and HstWeight. We also give the results by applying Hst to see the size of the cardinality-minimal solutions. See Table 2 for more details, where the sum of weights means the sum of weights of axioms in a solution and "TO" means time out.

From the table we can first observe that, Alg2 and HstSwoop have the same performance regarding the effectiveness. They can always find a solution with the minimal sum of weights. For instance, the sum of weights is 2.58 for Alg2 and HstSwoop while no less than 3.00 for other algorithms when dealing with ontology O_{m2}. Alg2 owes to the usage of the objective function. To find the solution with the minimal sum of weights, Alg2 may remove more axioms than Hst. Take O_{m4} as an example. Alg2 selected 3 axioms while Hst only chose 2 axioms, and the sum of weights is 1.02 and 2.00 respectively. Second, α in the fifth column indicates the minimal effective threshold determined by the minmax measure. From these thresholds we know that the axioms in most of the merged ontologies have higher weights. As for the correctness, all solutions have been checked and are correct. It verifies that using minmax measure could ensure Alg3 is able to find a solution.

As for the efficiency, HstWeight performs slightly better than ours since each extracted subset often contains one or two axioms and the search space has been reduced largely. Hst and HstSwoop achieved good performance for those ontologies with highly overlapping MUPS. For example, O_{m3} contains 685 MUPS while its solutions only include two or three axioms. For such ontologies, the HST-based algorithms are very efficient and can finish the process within 0.05 seconds. However, Hst and HstSwoop cannot deal with the ontologies once there are too many MUPS. This can be seen from the results of O_{r3}.

TABLE 2. The experimental results for ontologies with weights.

O	$\|O\|$	UC	MUPS	Algorithm	Solution Num	Solution Sum	Time (ms)
O_{m1}	529	5	17	Alg2	2	**1.75**	267
				Alg3($\alpha = 0.98$)	3	2.48	200
				HstSwoop	2	1.75	20
				HstWeight	3	2.48	4
				Hst	2	2.00	10
O_{m2}	433	14	37	Alg2	3	**2.58**	169
				Alg3($\alpha = 0.98$)	4	3.30	246
				HstSwoop	3	2.58	50
				HstWeight	4	3.30	4
				Hst	3	3.00	10
O_{m3}	877	17	685	Alg2	2	**1.52**	306
				Alg3($\alpha = 0.80$)	2	1.52	416
				HstSwoop	2	1.52	20
				HstWeight	3	2.31	14
				Hst	2	1.52	56
O_{m4}	861	32	167	Alg2	3	**1.02**	202
				Alg3($\alpha = 0.39$)	3	1.02	275
				HstSwoop	3	1.02	10
				HstWeight	4	1.38	6
				Hst	2	2.00	12
O_{m5}	359	32	3	Alg2	1	**0.31**	233
				Alg3($\alpha = 0.31$)	1	0.31	166
				HstSwoop	1	0.31	0
				HstWeight	2	0.55	2
				Hst	1	1.00	3
O_{m6}	449	7	7	Alg2	2	**1.95**	190
				Alg3($\alpha = 0.95$)	3	2.67	279
				HstSwoop	2	1.95	10
				HstWeight	3	2.67	10
				Hst	2	1.95	2
O_{m7}	357	18	24	Alg2	1	**0.91**	220
				Alg3($\alpha = 0.91$)	1	0.91	150
				HstSwoop	1	0.91	0
				HstWeight	1	0.91	0
				Hst	1	1.00	2
O_{m8}	450	3	10	Alg2	1	**0.68**	239
				Alg3($\alpha = 0.51$)	2	1.03	200
				HstSwoop	1	0.68	10
				HstWeight	2	1.03	0
				Hst	1	0.68	2
O_{m9}	544	9	8	Alg2	2	**1.66**	180
				Alg3($\alpha = 0.68$)	4	2.50	140
				HstSwoop	2	1.66	0
				HstWeight	5	3.08	0
				Hst	2	1.66	2
O_{r3}	5000	10	1620	Alg2	11	2.37	442
				Alg3($\alpha = 0.67$)	12	2.85	786
				HstSwoop	-	-	TO
				HstWeight	26	5.87	84
				Hst	-	-	TO

VI. RELATED WORK

We compare our work with a range of existing works on resolving logical contradictions.

To resolve logical contradictions, Reiter's Hitting Set Tree (HST) algorithm [10] is often used to satisfy the minimal change principle. The works given in [7], [8], and [29] apply the HST algorithm to the minimal unsatisfiability-preserving subsets for resolving incoherence. The work given in [9] applies the HST algorithm to the minimal inconsistent subsets

for resolving inconsistency. The work in [8] proposes various strategies to compute weights for axioms and then modifies the HST algorithm to find a solution with minimal sum of weights. This algorithm is much more efficient than **Hst** because the normal optimality criteria of the minimal path length has been replaced by the minimal path rank and there may exist quite a lot paths that could be early terminated. This algorithm corresponds to **HstSwoop** in our experiments. The work in [30] provides a graph-based algorithm to compute conflicts and then apply the HST algorithm to find solutions.

To improve the efficiency of the HST-based algorithms, some works provide algorithms to reduce the search space and satisfy other kinds of minimal change definitions. Such an algorithm first extracts a subset from each conflict and then applies the HST algorithm to those subsets. The typical work is given in [5] which proposed two algorithms to deal with the ontology revision problem. One algorithm makes use of the scoring function to choose those axioms with the highest score from each contradiction. The other exploits the weights to select those axioms with the lowest weight. The two algorithms correspond to **HstScore** and **HstWeight**. However, the efficiency of these algorithms is still a problem if the extracted subsets are not small enough.

To further improve the efficiency, the researchers propose heuristic strategies to find an approximately minimal solution, or adopt new semantics to avoid constructing a HST. The algorithm given in [31] removes one axiom with the highest score from the MIPS and then choose another one from the left MIPS. This process is repeated until no MIPS left. Obviously, the solution consisting of all removed axioms could not be minimal. The work in [30] also provides a scoring function to select axioms for revising an ontology. There are several works that deal with inconsistency or incoherence for the tasks of ontology learning [12], [23] and ontology versions [13]. They do not compute conflicts, but prevent an axiom being added to a coherent or consistent ontology when the axiom causing (potential) incoherence or inconsistency. The paper of [11] provides an efficient algorithm through replacing the standard semantics in characterising the standard inference tasks of DL Lite with an alternative semantics called type semantics. In such a case, the computation of conflicts and their minimal hitting sets can be avoided. We did not compare our algorithms with this algorithm because it only works for DL-Lite ontologies while our algorithms can deal with any DL expressivity.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we first defined the ILP-based model to resolve logical contradictions and then provided three algorithms to realize the model. In specific, **Alg1** assumes all axioms in an ontology have the same importance and the optimization objective is defined as minimizing the sum of all variables. **Alg2** assumes the axioms may have different importance and each axiom in an ontology is associated with a weight. In its objective function, each variable takes the weight of an axiom as a coefficient. To keep the axioms with important weights, we proposed **Alg3** to only remove those axioms with weights no more than a threshold. We used minmax measure to compute a minimal effective threshold to ensure a solution could be found. Besides, we proved that **Alg1** computes a cardinality-minimal incision function, **Alg2** computes a minimal one and **Alg3** computes an incision function.

According to the experimental results, it can be obviously revealed that our algorithms have excellent efficiency. They can finish each process of finding a solution within 0.5 seconds while those HST algorithms spent more than 100 seconds in many cases. **HstScore** and **HstWeight** could perform very well if the subsets where the HST algorithm is applied to contain very few axioms and the search space becomes very small. As for the effectiveness, **Alg1** and **Hst** could always find those cardinality-minimal solutions. Although both **Alg2** and **HstSwoop** could find a solution with minimal sum of weights, **Alg2** performs much better with respect to the efficiency. All found solutions have been verified to be correct.

In the future, we plan to apply the ILP-based model to ontology debugging and diagnosing tasks in DL ontologies or knowledge graphs [6], [18], [32]. Since a hitting set tree is often constructed when finding all conflicts for an unsatisfiable concept or an incoherent ontology, the ILP-based model may be used to largely reduce the computation cost. Besides, the model could also be used to measure the inconsistency or incoherence for showing the number of solutions with various cardinality [33], [34]. Finally, we plan to provide a user-friendly graphical interface by integrating our proposed algorithms into RaDON [35], which is a tool to debug and repair inconsistent or incoherent ontologies.

REFERENCES

[1] A. Soylu, M. Giese, E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, and I. Horrocks, "Ontology-based end-user visual query formulation: Why, what, who, how, and which?" *Universal Access Inf. Soc.*, vol. 16, no. 2, pp. 435–467, 2017.

[2] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jiménez-Ruiz, M. Giese, M. G. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, and I. Horrocks, "OptiqueVQS: A visual query system over ontologies for industry," *Semantic Web*, vol. 9, no. 5, pp. 627–660, 2018.

[3] I. G. Husein, B. Sitohang, S. Akbar, and F. N. Azizah, "Comparisons of diagnosis in mapping repair systems," in *Proc. Int. Conf. Data Softw. Eng.*, Oct. 2017, pp. 1–6.

[4] D. Lembo, R. Rosati, V. Santarelli, D. F. Savo, and E. Thorstensen, "Mapping repair in ontology-based data access evolving systems," in *Proc. IJCAI*, Melbourne, VIC, Australia, 2017, pp. 1160–1166.

[5] G. Qi, P. Haase, Z. Huang, Q. Ji, J. Z. Pan, and J. Völker, "A kernel revision operator for terminologies—Algorithms and evaluation," in *Proc. ISWC*, Karlsruhe, Germany, 2008, pp. 419–434.

[6] J. Du, "Ranking diagnoses for inconsistent knowledge graphs by representation learning," in *Proc. JIST*, Awaji, Japan, 2018, pp. 52–67.

[7] S. Schlobach, "Diagnosing terminologies," in *Proc. AAAI*, Pittsburgh, PA, USA, Jul. 2005, pp. 670–675.

[8] A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau, "Repairing unsatisfiable concepts in OWL ontologies," in *Proc. ESWC*, Budva, Montenegro, 2006, pp. 170–184.

[9] J. Du and G. Qi, "Tractable computation of representative ABox repairs in description logic ontologies," in *Proc. KSEM*, Chongqing, China, 2015, pp. 28–39.

[10] R. Reiter, "A theory of diagnosis from first principles," *Artif. Intell.*, vol. 32, no. 1, pp. 57–95, 1987.

[11] Z. Zhuang *et al.*, "DL-Lite contraction and revision," *J. Artif. Intell. Res.*, vol. 56, no. 1, pp. 329–378, 2016.
[12] J. Völker and M. Niepert, "Statistical schema induction," in *Proc. ESWC*, Heraklion, Greece, 2010, pp. 124–138.
[13] L. Bayoudhi, N. Sassi, and W. Jaziri, "How to repair inconsistency in OWL 2 DL ontology versions?" *Data Knowl. Eng.*, vol. 116, pp. 138–158, Jul. 2018.
[14] U. Straccia and F. Bobillo, "Mixed integer programming, general concept inclusions and fuzzy description logics," *Mathware Soft Comput.*, vol. 14, no. 3, pp. 247–259, 2007.
[15] F. Bobillo and U. Straccia, "Reducing the size of the optimization problems in fuzzy ontology reasoning," in *Proc. URSW ISWC*, Bethlehem, PA, USA, 2015, pp. 54–59.
[16] M. Niepert, C. Meilicke, and H. Stuckenschmidt, "A probabilistic-logical framework for ontology matching," in *Proc. AAAI*, Atlanta, GA, USA, 2010, pp. 1413–1418.
[17] J. Noessner, M. Niepert, C. Meilicke, and H. Stuckenschmidt, "Leveraging terminological structure for object reconciliation," in *Proc. ESWC*, Heraklion, Greece, 2010, pp. 334–348.
[18] Q. Wang, B. Wang, and L. Guo, "Knowledge base completion using embeddings and rules," in *Proc. IJCAI*, Buenos Aires, Argentina, 2015, pp. 1859–1866.
[19] A. Bate, B. Motik, B. C. Grau, D. T. Cucala, F. Simančík, and I. Horrocks, "Consequence-based reasoning for description logics with disjunctions and number restrictions," *J. Artif. Intell. Res.*, vol. 63, pp. 625–690, Nov. 2018.
[20] X. Zhang, J. Van den Bussche, and F. Picalausa, "On the satisfiability problem for SPARQL patterns," *J. Artif. Intell. Res.*, vol. 56, pp. 403–428, Jul. 2016.
[21] G. Qi, Q. Ji, J. Z. Pan, and J. Du, "Extending description logics with uncertainty reasoning in possibilistic logic," *Int. J. Intell. Syst.*, vol. 26, no. 4, pp. 353–381, 2011.
[22] S. Schlobach and R. Cornet, "Non-standard reasoning services for the debugging of description logic terminologies," in *Proc. IJCAI*, Acapulco, Mexico, 2003, pp. 355–362.
[23] P. Haase and J. Völker, "Ontology learning and reasoning—Dealing with uncertainty and inconsistency," in *Proc. URSW*, 2008, pp. 366–384.
[24] Q. Ji, Z. Gao, Z. Huang, and M. Zhu, "Measuring effectiveness of ontology debugging systems," *Knowl.-Based Syst.*, vol. 71, pp. 169–186, Nov. 2014.
[25] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. New York, NY, USA: Springer, 1972, pp. 85–103.
[26] Q. Ji, G. Qi, and P. Haase, "A relevance-directed algorithm for finding justifications of DL entailments," in *Proc. ASWC*, Shanghai, China, 2009, pp. 306–320.
[27] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler, "Swoop: A Web ontology editing browser," *J. Web Semantics*, vol. 4, no. 2, pp. 144–153, 2006.
[28] Q. Ji, Z. Gao, Z. Huang, and M. Zhu, "An efficient approach to debugging ontologies based on patterns," in *Proc. JIST*, Hangzhou, China, 2011, pp. 425–433.
[29] J. Du, G. Qi, J. Z. Pan, and Y.-D. Shen, "A decomposition-based approach to OWL DL ontology diagnosis," in *Proc. ICTAI*, Boca Raton, FL, USA, Nov. 2011, pp. 659–664.
[30] X. Fu, G. Qi, Y. Zhang, and Z. Zhou, "Graph-based approaches to debugging and revision of terminologies in DL-Lite," *Knowl.-Based Syst.*, vol. 100, pp. 1–12, May 2016.
[31] S. Schlobach, "Debugging and semantic clarification by pinpointing," in *Proc. ESWC*, Heraklion, Greece, 2005, pp. 226–240.
[32] J. Du, K. Qi, and Y. Shen, "Knowledge graph embedding with logical consistency," in *Proc. CCL*, Changsha, China, 2018, pp. 123–135.
[33] M. Thimm, "On the expressivity of inconsistency measures," *Artif. Intell.*, vol. 234, pp. 120–151, May 2016.
[34] H. Gao, J. Shi, G. Qi, and M. Wang, "Triple context-based knowledge graph embedding," *IEEE Access*, vol. 6, pp. 58978–58989, 2018.
[35] Q. Ji, P. Haase, G. Qi, P. Hitzler, and S. Stadtmüller, "RaDON—Repair and diagnosis in ontology networks," in *Proc. ESWC*, Heraklion, Greece, 2009, pp. 863–867.