Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 134950, 7 pages
http://dx.doi.org/10.1155/2015/134950

Research Article

A Max-Term Counting Based Knowledge Inconsistency Checking Strategy and Inconsistency Measure Calculation of Fuzzy Knowledge Based Systems

Hui-lai Zhi

School of Computer Science and Technology, Henan Polytechnic University, No. 2001, ShiJi Avenue, Jiaozuo 454000, China

Correspondence should be addressed to Hui-lai Zhi; zhihuilai@126.com

Received 27 April 2015; Revised 21 July 2015; Accepted 4 August 2015

Academic Editor: Miguel A. Salido

The task of finding all the minimal inconsistent subsets plays a vital role in many theoretical works especially in large knowledge bases and it has been proved to be a NP-complete problem. In this work, at first we propose a max-term counting based knowledge inconsistency checking strategy. And, then, we put forward an algorithm for finding all minimal inconsistent subsets, in which we establish a Boolean lattice to organize the subsets of the given knowledge base and use leaf pruning to optimize the algorithm efficiency. Comparative experiments and analysis also show the algorithm's improvement over past approaches. Finally, we give an application for inconsistency measure calculation of fuzzy knowledge based systems.

1. Introduction

A large knowledge system operating for a long time almost inevitably becomes polluted by wrong data that make the system inconsistent. Despite this fact, a sizeable part of the system remains unpolluted and retains useful information. It is widely adopted that a maximal consistent subset of a system contains a significant portion of unpolluted data [1]. So, simply characterizing a knowledge base as either consistent or inconsistent is of little practical value, and thus ensuring the consistency becomes an important issue [2–4].

In practice, there are two types of methods: one method is based on minimal inconsistent subsets, where every strict subset is consistent and the other is directly based on maximal consistent subsets. Actually, the relationship between minimal inconsistent subsets and maximal consistent subsets was discovered separately in [1, 5, 6], which is known as the hitting subset problem [7].

As finding minimal inconsistent subsets or maximal consistent subsets is NP-complete, the most efficient algorithm is not known yet, and there are a number of heuristic optimizations that can be used to substantially reduce the size of the search space. In practice, heuristic information [8, 9], optimization [10, 11], and hybrid techniques [12, 13] are recognized to reduce time complexity. In the latest research, McAreavey et al. presented a computational approach to finding and measuring inconsistency in arbitrary knowledge bases [14], while Mu et al. gave a method for measuring the significance of inconsistency in the viewpoints framework [15]. In all the abovementioned works, effectively finding minimal inconsistent subsets is the critical step which has a great impact on the applications especially for large knowledge bases. Apparently, its computational complexity depends on the underlying strategies used for checking the consistency of subsets of the knowledge base, but till now this important issue has not gotten a satisfying solution.

In this paper, we first propose an efficient strategy to check the consistency of a given knowledge base. And, then, we put forward an algorithm to find all of the minimal inconsistent subsets of the given knowledge base. Thereafter, to illustrate the algorithm's improvement, we conduct thorough comparative experiments and analysis with respect to one of the latest proposed algorithm MARCO [16] and give a discussion on the relative algorithms DAA [17] and PDDS [18]. Finally, we give an application for inconsistency measure calculation of fuzzy knowledge based systems.

2. Theoretical Basis

Let L denote the propositional language built from a finite set of variables P using logical connectives $\{\wedge, \vee, \neg, \rightarrow\}$ and logical constants $\{T, F\}$. Every variable $p \in P$ is called an atomic formula or an atom. A literal is an atom or its negation. A clause φ is a formula restricted to a disjunction of literals and let $\mathrm{var}(\varphi)$ denote the set of variables in a clause φ. A knowledge base $K \in 2^L$ is a finite set of arbitrary formulae.

As every formula can be converted into an equivalent conjunction normal form (CNF) formula, knowledge base can be normalized in such a way that every formula contained in it is a clause. For a given normalized knowledge base, if there are no redundant clauses, we say it is an optimized knowledge base.

By the syntactic approach in proof theory, if both φ and $\neg\varphi$ can be derived from a knowledge base K, then we say K is inconsistent. With the semantic approach in model theory, an interpretation or world ω is a function $\omega : P \mapsto \{F, T\}$ from P to the set of Boolean values $\{F, T\}$. Let 2^P denote the set of worlds of L. A world ω is a model of K, denoted as $\omega \Rightarrow K$, iff K is true under ω in the classical truth-functional manner. Let $\mathrm{mod}(K)$ denote the set of models of K; that is, $\mathrm{mod}(K) = \{\omega : \omega \in 2^P \mid \omega \Rightarrow K\}$. We say that K is satisfiable iff there exists a model of K. Conversely, K is unsatisfiable iff there are no models of K. These two approaches coincide in propositional logic; that is, a knowledge base K is consistent iff K is satisfiable.

In the following discussion, let the Greek lower case letters $\varphi, \psi, \ldots$ be formulae from L and English lower case letters $a, b, \ldots$ variables from P.

Definition 1. For a Boolean function of n variables $x_1, \ldots, x_n$, a sum term in which each of the n variables appears once (in either its complemented or uncomplemented form) is called a max-term.

Proposition 2. *Let $x_1, \ldots, x_n$ be n variables and $M_0, \ldots, M_{2^n-1}$ the 2^n different max-terms built on these n variables. Then $\prod_{i=0}^{2^n-1} M_i = M_0 \wedge M_1 \wedge \cdots \wedge M_{2^n-1} = F$.*

For example, let $M_0 = p \vee q$, $M_1 = p \vee \neg q$, $M_2 = \neg p \vee q$, and $M_3 = \neg p \vee \neg q$ be formulas built on $\{p, q\}$. Then it is trivial to show that $M_0 \wedge M_1 \wedge M_2 \wedge M_3 = F$, which means no assignments to p and q satisfy M_1, M_2, M_3, and M_4 simultaneously.

Definition 3. Let φ be a formula built on a set of variables P. Then we call $\mathrm{ext}(\varphi)$ the extension of φ; that is,

$$\mathrm{ext}(\varphi) = \left\{\varphi \vee a_1^{\beta_1} \vee \cdots \vee a_s^{\beta_s}\right\}, \quad (\beta_1, \ldots, \beta_s) \in \{0,1\}^s, \ \{a_1, \ldots, a_s\} = P - \mathrm{var}(\varphi), \tag{1}$$

and, for each $i \in \{1, 2, \ldots s\}$, $a_i^0 = a_i$, $a_i^1 = \neg a_i$.

For example, let $\varphi = p \vee \neg q$ be a formula built on $P = \{p, q, r\}$. Then we have $\mathrm{var}(\varphi) = \{p, q\}$ and $\mathrm{ext}(\varphi) = \{p \vee \neg q \vee r, p \vee \neg q \vee \neg r\}$. Actually, a simple manipulation leads to $\varphi = p \vee \neg q = (p \vee \neg q \vee r) \wedge (p \vee \neg q \vee \neg r)$.

Remark 4. It is easy to see that carrying extension of φ does not change the original meaning of φ.

Theorem 5. *Let $\varphi_1 = a_1^{\beta_1} \vee a_2^{\beta_2} \vee \cdots \vee a_r^{\beta_r}$ ($\beta_i \in \{0,1\}$ ($i \in \{1, 2, \ldots, r\}$)) and $\varphi_2 = a_1^{\beta_1'} \vee a_2^{\beta_2'} \vee \cdots \vee a_s^{\beta_s'}$ ($\beta_j' \in \{0,1\}$ ($j \in \{1, 2, \ldots, s\}$)) be two formulas built on a set of variables $P(|P| = m)$, in which $a_i^0 = a_i$, $a_i^1 = \neg a_i$. Then the following propositions hold:*

(i) *If there exists a variable a_i such that one of a_i and $\neg a_i$ appears in φ_r and the other one appears in φ_s, then $|\mathrm{ext}(\varphi_1) \cap \mathrm{ext}(\varphi_2)| = 0$.*

(ii) *Otherwise, $|\mathrm{ext}(\varphi_1) \cap \mathrm{ext}(\varphi_2)| = 2^{m-|\mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2)|}$.*

Proof. (i) As there exists a variable a_i such that one of a_i and $\neg a_i$ appears in φ_1 and the other one appears in φ_2, then one of a_i and $\neg a_i$ must appear in $\mathrm{ext}(\varphi_1)$ and the other one must appear in $\mathrm{ext}(\varphi_2)$, which makes $\mathrm{ext}(\varphi_1)$ differ from $\mathrm{ext}(\varphi_2)$. Therefore we have $|\mathrm{ext}(\varphi_1) \cap \mathrm{ext}(\varphi_2)| = 0$.

(ii) If there does not exist a variable a_i such that one of a_i and $\neg a_i$ appears in φ_1 and the other one appears in φ_2, then there are two situations that need to be surveyed, respectively.

Situation 1. If $\mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2) = P$, then there exists only one common formula of φ_1 and φ_2, that is, $\varphi_1 \vee \varphi_2$. Hence, we have $|\mathrm{ext}(\varphi_1) \cap \mathrm{ext}(\varphi_2)| = 1$, which is equivalent to

$$\left|\mathrm{ext}\left(\varphi_1\right) \cap \mathrm{ext}\left(\varphi_2\right)\right| = 2^{m-|\mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2)|} = 2^0 = 1. \tag{2}$$

Situation 2. If $\mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2) \subset P$, then the common formula of φ_1 and φ_2 is

$$\varphi_1 \vee \varphi_2 \vee a_1^{\beta_1} \vee \cdots \vee a_s^{\beta_s} \quad \left(\beta_i \in \{0,1\} \ (i \in \{1, 2, \ldots, s\})\right), \ a_i^0 = a_i, \ a_i^1 = \neg a_i, \tag{3}$$

where $\{a_1, \ldots, a_s\} = P - \mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2)$. Hence, we have

$$\left|\varphi_1 \vee \varphi_2 \vee a_1^{\beta_1} \vee \cdots \vee a_s^{\beta_s}\right| = 2^{m-|\mathrm{var}(\varphi_1) \cup \mathrm{var}(\varphi_2)|}. \tag{4}$$

Therefore, combining the above two situations, the theorem is proved. □

Corollary 6. *Let $\{\varphi_1, \varphi_2, \ldots, \varphi_k\}$ be a set of formulas built on a set of variables $P(|P| = m)$. Then the following propositions hold:*

(1) *If there exist a variable a_i and two formulas φ_r and φ_s such that one of a_i and $\neg a_i$ appears in φ_r and the other one appears in φ_s, then $|\bigcap_{i=1}^k \mathrm{ext}(\varphi_i)| = 0$.*

(2) *Otherwise, $|\bigcap_{i=1}^k \mathrm{ext}(\varphi_i)| = 2^{m-|\bigcup_{i=1}^k \mathrm{var}(\varphi_i)|}$.*

Theorem 7. *Let $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ be a set of formulas built on a set of variables $P(|P| = m)$. Then after carrying*

extensions for $\varphi_1, \varphi_2, \ldots, \varphi_n$, respectively, and using σ to denote the number of different formulas obtained, we have

$$\begin{aligned}\sigma &= \left|\bigcup_{i=1}^{n} \text{ext}(\varphi_i)\right| \\ &= \sum_{i=1}^{n} \left|\text{ext}(\varphi_i)\right| - \sum_{1\le i<j\le n} \left|\text{ext}(\varphi_i) \cap \text{ext}(\varphi_j)\right| + \cdots \\ &\quad + (-1)^{n+1} \left|\bigcap_{i=1}^{k} \text{ext}(\varphi_i)\right|.\end{aligned} \tag{5}$$

Moreover, if $\sigma = 2^m$, then Φ is inconsistent; otherwise Φ is consistent.

Proof. According to inclusion-exclusion principle and in light of Proposition 2 the proof is trivial. □

Let $\Phi = \{p, \neg p \vee q, r\}$ be a set of formulas defined on $\{p, q, r\}$. According to Theorem 7, we have $\sigma = 2^{3-1} + 2^{3-2} + 2^{3-2} - 0 - 2^{3-2} - 2^{3-3} + 0 = 5$ and thus Φ is consistent.

3. An Algorithm for Finding All Minimal Inconsistent Subsets

In this section, at first we propose an algorithm for finding all nominal inconsistent subsets via Boolean lattice. And, then, we give an illustrative example and a thorough comparative study with algorithm MARCO by using the number of visited subsets as the benchmark. Besides this, we also give a discussion on relative algorithms DAA and PDDS.

3.1. Algorithm. An algorithm to find the minimal inconsistent subsets of a given knowledge system must check each of its subsets for inconsistency. One way to proceed is to construct a Boolean lattice of subsets of the given knowledge system, which is initially used by Bird and Hinze in the process of finding the maximal consistent subsets [19].

A lattice L is called a Boolean lattice if

(i) L is distributive,

(ii) L has 0 and 1,

(iii) each $a \in L$ has a complement $a^* \in L$.

Figure 1 sketches a three-variable Boolean lattice, where all the labels of the nodes consist of the power set of set $\{a, b, c\}$.

In Algorithm 1, a Boolean lattice is also established and leaf pruning is adopted to optimize the algorithm efficiency. Because the cardinality of the subsets at each level is smaller than those on the level above it, a breadth-first search of the lattice will consider all smaller sets before any larger ones. Apparently, leaf pruning strategy can be used based on the fact that if a node denotes an minimal inconsistent subset, then all of its ancestors are inconsistent; dually, if a node denotes a consistent subset, then all of its descendents are consistent.

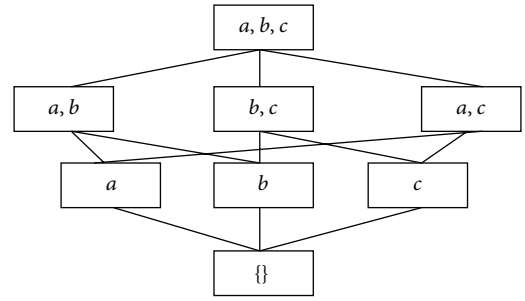

Figure 1: Three-variable Boolean lattice.

According to Theorem 7, directly computing σ is time consuming, so we will store the intermediate calculation results. For example, in Algorithm 1, for each visited node (whose corresponding formula set is $\{\varphi_i \mid i \in T\}$), we will store the value of $|\bigcap_{i\in T} \text{ext}(\varphi_i)|$. Apparently, when computing the σ value of k-degree node, the stored value $|\bigcap_{i\in T} \text{ext}(\varphi_i)|$ of each $(k-1)$-degree nodes can be reused to save time cost.

In the minimal inconsistent subsets finding algorithm proposed in [14], which is derived indirectly on maximal consistent subset, there exists a disadvantage that while getting maximal consistent subset, pseudo-maximal consistent subset will be generated [11, 15]. As our proposed algorithm always checks the smaller sets before the larger sets, so it can overcome this problem.

If the maximal cost for checking inconsistency of is T, the complexity of this algorithm is $O(2^{|K|}T)$ in the worst case. In the following, by using experiment, we will show the relationship between the number of subsets that were checked for inconsistency and the size the of the knowledge base with respect to different probabilities that two formulas are consistent.

In the experiment, we use generator GENBAL [20] to generate knowledge bases. The graphs in Figures 2 and 3 show the number of subsets that were checked for inconsistency related to $|K|$, the number of clauses contained in the given normalized knowledge base and p, and the probability that two formulas are consistent. All counts are averaged across 100 randomly generated formulae by using GENBAL.

From Figures 2 and 3, we can see that larger values for p mean that more subsets will be checked. Moreover, it is easy to show that larger values for p generally also lead to fewer and smaller minimal inconsistent subsets.

3.2. An Illustrative Example and Comparative Study. Apart from our proposed method, there are many other solvers for computing minimal inconsistent subsets. One of the latest published algorithm is MARCO [16], which adopts the most recent advances. At first we give an illustrative example and then we compare our method with MARCO.

Example 8. Considering a set of formulas, $1 : (a), 2 : (\neg a), 3 : (\neg a \vee b)$, and $4 : (\neg b)$, which is used in [16], we use Algorithm 1 to find all the minimal inconsistent subsets.

At first we also establish a Boolean lattice, which is shown in Figure 4.

Input: a knowledge base K
Output: all the minimal inconsistent subsets MI(K) of knowledge base K
Begin
(1) normalize the knowledge base K to ensure that every formula contained in it is a clause,
and denoted it as norm(K) = $\{\varphi_1, \varphi_2, \ldots, \varphi_m\}$;
initialize MI(K) to be an empty set of sets;
(2) build a m variable Boolean lattice BL_m with each node denoting a set of formulas
(if a node denotes a n formula set, then we call it a n-degree node), and give each node a unmarked flag;
(3) set up an empty list *List*, and put all of the 2-degree nodes into *List*;
fetch the head *Head* from *List*;
if *Head* is inconsistent, then //by using Theorem 7
Begin
put *Head* into MI(K);
mark all the ancestors of *Head*;
End
else *Head* is consistent, then
Begin
insert all of the un-marked upper neighbors of *Head* at the front of *List*;
mark all the descendants of *Head*, and if they exist in *List*, then remove them from *List*;
End
(4) return MI(K).
End.

Algorithm 1: Finding all the minimal inconsistent subsets of a knowledge base.

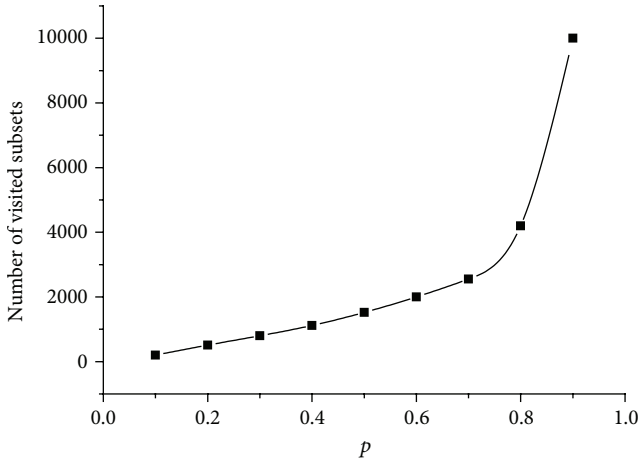

Figure 2: Number of subsets visited as a function of p for $|K| = 15$.

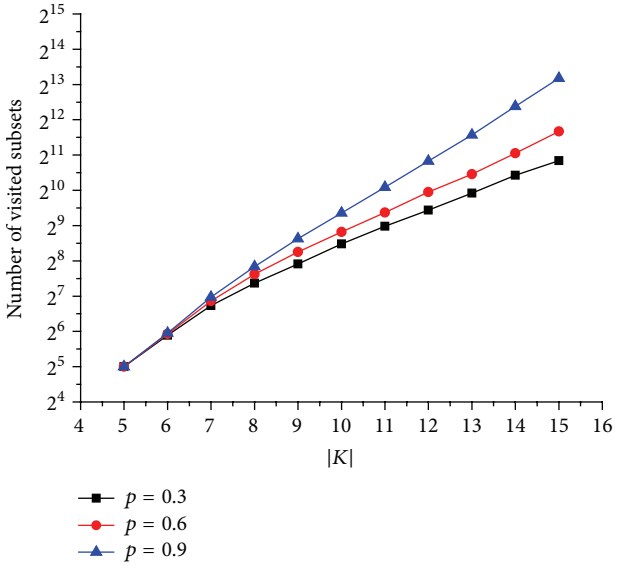

Figure 3: Number of subsets visited as a function of $|K|$ for $p = 0.3$, $p = 0.6$, and $p = 0.9$.

Then we establish a list *List*, and initialize *List* as

$$List : \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}. \quad (6)$$

Fetch the head *Head* from *List*; that is, *Head* = $\{1,2\}$. According to Theorem 7, *Head* = $\{1,2\}$ is inconsistent, and then mark its ancestors $\{1,2,3\}$, $\{1,2,4\}$, and $\{1,2,3,4\}$.

Fetch the head *Head* from *List*; that is, *Head* = $\{1,3\}$. According to Theorem 7, *Head* = $\{1,3\}$ is consistent; insert its unmarked upper neighbor $\{1,3,4\}$ at the front of *List*, and mark its descendants $\{1\}$ and $\{3\}$.

Fetch the head *Head* from *List*; that is, *Head* = $\{1,3,4\}$. According to Theorem 7, *Head* = $\{1,3,4\}$ is inconsistent.

Fetch the head *Head* from *List*; that is, *Head* = $\{1,4\}$. According to Theorem 7, *Head* = $\{1,4\}$ is consistent, and mark its unmarked descendant $\{4\}$.

Fetch the head *Head* from *List*; that is, *Head* = $\{2,3\}$. According to Theorem 7, *Head* = $\{1,3\}$ is consistent; insert its unmarked upper neighbor $\{2,3,4\}$ at the front of *List*, and mark its unmarked descendant $\{2\}$.

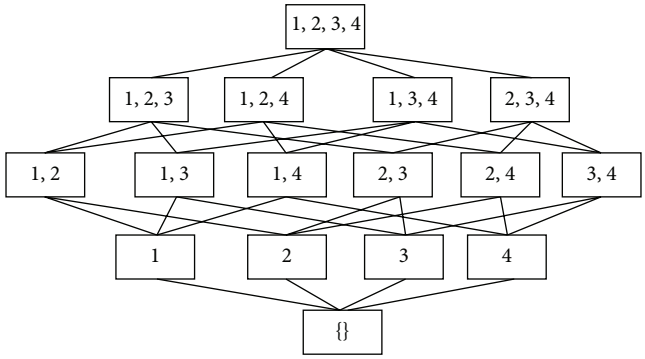

Figure 4: Four-variable Boolean lattice.

Fetch the head *Head* from *List*; that is, *Head* = {2, 3, 4}. According to Theorem 7, *Head* = {1, 3} is consistent, and mark its unmarked descendant {2, 4} and {3, 4} and remove {2, 4} and {3, 4} from *List*.

At this point, *List* is empty; algorithm terminates with results {1, 2} and {1, 3, 4}.

It is apparent that in order to get all the minimal inconsistent subsets we have to judge the consistency of 6 sets, which are

$$\{1,2\},\{1,3\},\{1,3,4\},\{1,4\},\{2,3\},\{2,3,4\}. \quad (7)$$

Fundamentally, the MARCO algorithm operates repeatedly:

(i) Selecting an unexplored point in the power set lattice, a subset of C that we call a seed.

(ii) Checking the satisfiability of the seed.

(iii) Growing or shrinking it to an MSS or an MUS as appropriate.

(iv) Marking a corresponding region of the lattice as explored.

When we use algorithm MARCO, the consistency of 10 sets needs to be considered one by one, which are

$$\{1,2,4\},\{1,2\},\{3\},\{1,3\},\{2\},\{2,3\},\{2,4\},\{2,3,4\},\{1,4\},\{1,3,4\}. \quad (8)$$

So, our algorithm performs better than MARCO with respect to the number of the visited sets.

The difference between our algorithm and MARCO is that our algorithm traverses the Boolean lattice incremental according to the cardinalities of the sets while MARCO traverses the Boolean lattice randomly, as the function *GetUnexplored*() used in MARCO randomly returns any unexplored sets. As the objectives of both algorithms are to find all of the minimal sets which are inconsistent, the incremental feature of our algorithm brings a higher efficiency.

In the comparative study, we also use generator GENBAL [20] to generate knowledge base. Number of visited subsets is used as the benchmarks, as it is more objective than the other benchmarks. For example, we do not choose CPU times as the benchmark as it is strongly affected by the running environment, including the status of both hardware and software.

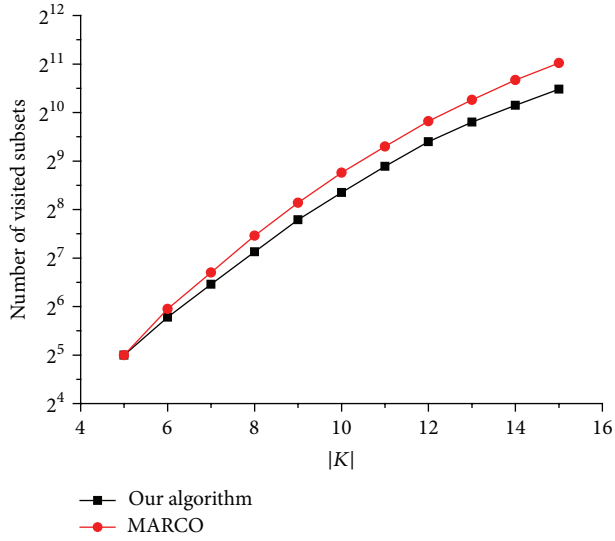

Figure 5: Number of subsets visited as a function of $|K|$ for $p = 0.3$.

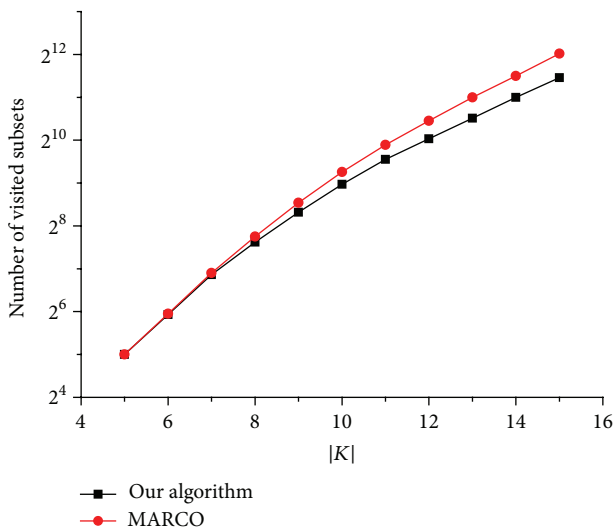

Figure 6: Number of subsets visited as a function of $|K|$ for $p = 0.6$.

The graphs in Figures 5, 6, and 7 show the number of subsets that were checked for inconsistency related to $|K|$, the number of clauses contained in the given normalized knowledge base and p, and the probability that two formulas are consistent. All counts are averaged across 100 randomly generated formulae by using GENBAL. All of Figures 5, 6, and 7 show that our algorithm performs better than MARCO with respect to the number of the visited sets.

DAA is another algorithm that exploits the hitting set duality between minimal correction sets (MCSes) and minimal unsatisfiable subsets [17]. DAA uses the Grow subroutine on known-satisfiable subsets to produce maximal satisfiable subsets (MSSes) and their complementary MCSes and then computes minimal hitting sets of the MCSes found thus

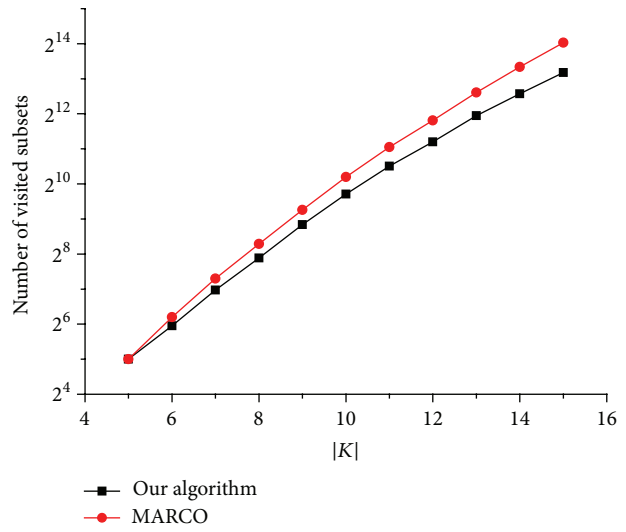

Figure 7: Number of subsets visited as a function of $|K|$ for $p = 0.9$.

far. PDDS, an approach closely related to DAA, was later proposed [18]. The main differences are that PDDS takes an initial set of either maximal satisfiable subsets (MUSes) or MCSes as input, and PDDS does not necessarily compute all hitting sets of the MCSes at each iteration, avoiding the memory scaling issues of DAA.

The DAA and PDDS algorithms have the benefit that they are decoupled from the choice of hitting set algorithm. It is pointed out that the choice of the incremental algorithm presented by Fredman and Khachiyan [21] for computing hitting sets results in a version of the DAA algorithm with worst case runtime that is subexponential in the size of the output [22]. And studies have shown that MARCO performs better than DAA and PDDS [16].

4. Inconsistency Measure Calculation for Fuzzy Knowledge Based Systems

In this section, we show an application of Algorithm 1 for inconsistency measure calculation of fuzzy knowledge based systems.

Fuzzy knowledge based systems are a typical rule-based inference system for providing expertise over a domain, which is capable of drawing conclusions from given uncertain evidence [23]. In fuzzy knowledge based systems, knowledge is represented by using possibilistic logic.

Let $\Delta = \{(\varphi_1, \alpha_1), (\varphi_2, \alpha_2), \ldots, (\varphi_n, \alpha_n)\}$ denote a fuzzy knowledge based system, in which $\varphi_1, \varphi_2, \ldots, \varphi_n$ are classical propositional logic formulas and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are their possibility measures.

Definition 9. Let Δ be a fuzzy knowledge based system. If Δ contains two formulas (φ, α_1), (φ, α_2) with $\alpha_1 > \alpha_2$, then we call $\Delta' = \Delta - (\varphi, \alpha_2)$ the possibility based deduction result of Δ and denote by $d(\Delta) = \Delta'$.

Definition 10. Let (φ, α) be a possibility formula built on a set of variables P. Then we call $\text{ext}(\varphi, \alpha)$ the extension of (φ, α); that is,

$$\text{ext}(\varphi, \alpha) = \left\{\varphi \vee \gamma_1^{\beta_1} \vee \cdots \vee \gamma_s^{\beta_s}, \alpha\right\}, \quad (\beta_1, \ldots, \beta_s) \in \{0,1\}^s, \ \{\gamma_1, \ldots, \gamma_s\} = P - \text{var}(\varphi), \tag{9}$$

and for each $i \in \{1, 2, \ldots, s\}$, $\gamma_i^0 = \gamma_i$, $\gamma_i^1 = \neg\gamma_i$.

Definition 11. Let $\Delta = \{(\varphi_1, \alpha_1), (\varphi_2, \alpha_2), \ldots, (\varphi_n, \alpha_n)\}$ be a fuzzy knowledge based system. Then we call $\pi(\Delta) = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ the projection of Δ onto the classical knowledge base.

Definition 12 (see [24]). Let Δ be a fuzzy knowledge based system. If Δ is inconsistent, then its inconsistency measure is defined as

$$\text{INCON}(\Delta) = \max_{\Delta' \subseteq \Delta, \text{INCON}(\Delta') > 0} \min\left\{\alpha \mid (\varphi, \alpha) \in \Delta'\right\}. \tag{10}$$

Theorem 13. *Let Δ be a fuzzy knowledge based system built on a set of variables $P(|P| = m)$. After extensions and possibility based deduction of Δ are performed, Δ' is derived. If $|\pi(\Delta')| = 2^m$, then Δ is inconsistent, and its inconsistency measure is*

$$\text{INCON}(\Delta) = \text{Min}\left\{\alpha_i \mid (\varphi_i, \alpha_i) \in \Delta'\right\}. \tag{11}$$

Proof. As $\text{INCON}(\Delta) = \max_{\Delta^* \subseteq \Delta', \text{INCON}(\Delta^*) > 0} \min\{\alpha \mid (\varphi, \alpha) \in \Delta^*\}$ and Δ' is result of extensions and possibility based deduction of Δ, then $\text{INCON}(\Delta) = \text{INCON}(\Delta')$. Since $|\pi(\Delta')| = 2^m$, we know that Δ' is inconsistent, and its inconsistency measure is $\text{Min}\{\alpha_i \mid (\varphi_i, \alpha_i) \in \Delta'\}$. Hence, the theorem is proved. □

Example 14. Let $\Delta = \{(\neg p \vee q, 0.6), (p \vee \neg q, 0.8), (q \vee r, 1), (r, 0.3), (\neg r, 0.5)\}$ be a fuzzy knowledge based system built on $\{p, q, r\}$.

After carrying extensions and possibility based deduction of Δ, we get

$$\begin{aligned}\Delta' = \{&(\neg p \vee q \vee \neg r, 0.6), (p \vee \neg q \vee r, 0.8),\\ &(p \vee \neg q \vee \neg r, 0.8), (p \vee q \vee r, 1), (\neg p \vee q \vee r, 1),\\ &(\neg p \vee \neg q \vee r, 0.3), (p \vee q \vee \neg r, 0.5),\\ &(\neg p \vee \neg q \vee \neg r, 0.5)\}.\end{aligned} \tag{12}$$

According to Theorem 13, we know that Δ is inconsistent and its inconsistency measure is 0.3.

5. Conclusions

The purpose of this paper is to find all the minimal inconsistent subsets of a given knowledge system. Initially we propose a max-term counting based knowledge inconsistency checking strategy. And, then, we put forward an algorithm for finding all minimal inconsistent subsets, in which we establish a Boolean lattice to organize the subsets of the given

knowledge base and use leaf pruning to optimize the algorithm efficiency. Finally, we give a method for inconsistency measure calculation of fuzzy knowledge based system.

As in a fuzzy knowledge based system, there may be several statements in contradiction to each other; how to measure the significance of the inconsistency is a valuable problem for further study.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The work presented in this paper is supported by Doctorial Foundation of Henan Polytechnic University (B2011-102). The author also gratefully acknowledges the helpful comments and suggestions of the reviewers, which have greatly improved the presentation.

References

[1] E. Birnbaum and E. L. Lozinskii, "Consistent subsets of inconsistent systems: structure and behaviour," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 15, no. 1, pp. 25–46, 2003.

[2] Z. Sun, Z. Zhang, and H. Wang, "Consistency and error analysis of prior-knowledge-based kernel regression," *Neurocomputing*, vol. 74, no. 17, pp. 3476–3485, 2011.

[3] J. Ramírez and A. de Antonio, "Checking the consistency of a hybrid knowledge base system," *Knowledge-Based Systems*, vol. 20, no. 3, pp. 225–237, 2007.

[4] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, "Consistency-based diagnosis of configuration knowledge bases," *Artificial Intelligence*, vol. 152, no. 2, pp. 213–234, 2004.

[5] J. Bailey and P. J. Stuckey, "Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization," in *Proceedings of the 7th International Symposium on Practical Aspects of Declarative Languages (PADL '05)*, pp. 174–186, January 2005.

[6] M. H. Liffiton, M. D. Moffitt, M. E. Pollack, and K. A. Sakallah, "Identifying conflicts in overconstrained temporal problems," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 205–211, August 2005.

[7] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, E. R. Miller and J. W. Thatcher, Eds., pp. 85–103, Plenum Press, New York, NY, USA, 1972.

[8] I. Shah, "A hybrid algorithm for finding minimal unsatisfiable subsets in over-constrained CSPs," *International Journal of Intelligent Systems*, vol. 26, no. 11, pp. 1023–1048, 2011.

[9] I. Shah, "Direct algorithms for finding minimal unsatisfiable subsets in over-constrained CSPs," *International Journal on Artificial Intelligence Tools*, vol. 20, no. 1, pp. 53–91, 2011.

[10] A. Felfernig, M. Schubert, and C. Zehentner, "An efficient diagnosis algorithm for inconsistent constraint sets," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 26, no. 1, pp. 53–62, 2012.

[11] R. Malouf, "Maximal consistent subsets," *Computational Linguistics*, vol. 33, no. 2, pp. 153–160, 2007.

[12] E. Di Rosa, E. Giunchiglia, and M. Maratea, "Solving satisfiability problems with preferences," *Constraints*, vol. 15, no. 4, pp. 485–515, 2010.

[13] M. H. Liffiton and K. A. Sakallah, "Algorithms for computing minimal unsatisfiable subsets of constraints," *Journal of Automated Reasoning*, vol. 40, no. 1, pp. 1–33, 2008.

[14] K. McAreavey, W. Liu, and P. Miller, "Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases," *International Journal of Approximate Reasoning*, vol. 55, no. 8, pp. 1659–1693, 2014.

[15] K. Mu, Z. Jin, W. Liu, D. Zowghi, and B. Wei, "Measuring the significance of inconsistency in the viewpoints framework," *Science of Computer Programming*, vol. 78, no. 9, pp. 1572–1599, 2013.

[16] M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva, "Fast, flexible MUS enumeration," *Constraints*, 2015.

[17] J. Bailey and P. J. Stuckey, "Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization," in *Proceedings of the 7th International Symposium on Practical Aspects of Declarative Languages (PADL '05)*, vol. 3350, pp. 174–186, January 2005.

[18] R. T. Stern, M. Kalech, A. Feldman, and G. M. Provan, "Exploring the duality in conflict-directed model-based diagnosis," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence and the 24th Innovative Applications of Artificial Intelligence Conference (AAAI '12)*, pp. 828–834, July 2012.

[19] R. Bird and R. Hinze, "Functional pearl: trouble shared is trouble halved," in *Proceedings of the ACM SIGPLAN Workshop on Haskell*, pp. 1–6, Uppsala, Sweden, 2003.

[20] J. A. Navarro and A. Voronkov, "Generation of hard non-clausal random satisfiability problems," in *Proceedings of the 20th National Conference on Artificial Intelligence*, pp. 436–442, July 2005.

[21] M. L. Fredman and L. Khachiyan, "On the complexity of dualization of monotone disjunctive normal forms," *Journal of Algorithms*, vol. 21, no. 3, pp. 618–628, 1996.

[22] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharma, "Discovering all most specific sentences," *ACM Transactions on Database Systems*, vol. 28, no. 2, pp. 140–174, 2003.

[23] E. Sanchez, *Approximate Reasoning in Intelligent Systems, Decision and Control*, Pergamon Press, 1st edition, 1987.

[24] J. Lang, "Possibilistic logic: complexity and algorithms," in *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, D. M. Gabby, P. Semts, J. Kohlas, and S. Moral, Eds., pp. 179–220, Springer, 1997.