为了方便检测节点对应的术语集关于概念的可满足性,将术语集 T 关于概念 C 的搜索图 $G(V,E,r)$的节点集合 V 分为 3 类:live 节点、dead 节点和 pending 节点.其定义分别为:live 节点表示其对应的术语集中并未包含互补相关公理,即该术语集关于概念 C 是可满足的,并且搜索过程正运行于当前节点的分支上,该类节点可以继续向下扩展;dead 节点表示其对应的公理集合为互补概念术语集,这类节点不能继续向下扩展,需要进行概念可满足性检测;pending 节点表示当前图中未被搜索过的节点.搜索图中 3 类节点之间的转换关系如下:当一个 pending 节点所对应的术语集为互补概念术语集时,该节点就转化为 dead 节点;当算法的搜索过程到达一个 pending 节点,并且其对应的术语集不包括互补相关公理时,就将该节点转化为 live 节点.

假设一个关于概念 C 不可满足的术语集 T,基于宽度优先搜索的子术语集扩展算法为术语集 T 构造一个关于 C 的搜索图 $G(V,E,r)$,形如 $C\sqsubseteq\top$的公理对应的节点为汇节点 r,对于搜索图 G 中的每个节点 $v\in V$,v 都对应术语集 T 的一个子集.基于宽度优先搜索的子术语集扩展算法主要思想如下:通过为输入的术语集 T 构造一个搜索图 $G(V,E,s)$,采用宽度优先搜索查找图中的 dead 节点.因为 dead 节点包含互补相关公理,需要判断其对应的术语集 T'是否满足概念 C.若 T'不满足 C,T'为不可满足子术语集,返回 T',否则,需要回溯到父节点,将使用选择函数生成的术语集作为待测术语集进行概念 C 可满足性检测.具体算法如下所示.

算法 3. 基于宽度优先搜索的子术语集扩展算法.

输入:不可满足概念 C,术语集 T;

输出:一个不可满足子术语集 T'.

```
1.  k←1,T'←∅,W=∅
2.  while W≠T do
3.    S←s(T,C,k)
4.    Σ←s(T,C,k+1)\s(T,C,k)
5.    if W=∅
6.      W←{S}
7.    for each axiom a_x∈Σ do
8.      for S'∈W do
9.        if Compl(S'∪{ax}) is false and S'∪{ax}∉W
10.         W←W{S'∪{ax}}
11.       if Compl(S'∪{ax}) is true
12.         if C is not satisfiable w.r.t. S'∪{ax}
13.           T'←S'∪{ax}
14.         else
15.           while C is satisfiable w.r.t. s(T,C,k+1)
16.                 k←k+1
17.           T'←s(T,C,k)
18.         return T'
19.   k←k+1
20. return T
```

算法 3 通过 3 层循环构造术语集 T 关于概念 C 的搜索图 $G(V,E,r)$,并以宽度优先搜索策略查找 $G(V,E,r)$中的 dead 节点.k 从 1 开始递归地产生与不可满足概念 C 直接或间接相关的公理集合,加入到 $s(T,C,k)$中.S 为第 k 次使用选择函数获得的公理集合,而Σ为第 k+1 次使用选择函数获得的新的公理集合.根据这两个变量生成搜索图 $G(V,E,r)$中节点对应的公理集合 S':初始化 S'为 $s(T,C,k)$,对于Σ中的任何一个公理 ax,如果 $S'\cup\{ax\}$是互补概念术语集,则表明该术语集对应的节点为 dead 节点,判断该术语集是否满足概念 C.若 $S'\cup\{ax\}$不满足 C,则 $S'\cup\{ax\}$作为结果返回,否则,将当前的 $s(T,C,k)$作为子术语集进行概念 C 的可满足性检测,这部分与算法 2 一致;S'

∪{ax}中未包含互补相关公理,说明该节点为 live 节点,需要进一步扩展,将 ax 加入到当前节点所在的术语集中作为下一轮循环中的父节点.算法 3 使用了如下启发式来优化搜索过程:(1) 第 8 行 W 中选择术语集 S'的过程中,以公理元素个数对 W 中的公理集合进行排序,从元素个数最少的术语集开始选择;(2) 第 7 行选择公理 ax 的过程中,对Σ中的公理进行排序,将包含原子概念否定的公理排在其他公理之前.

图 2 表示 T_1 中关于概念 A_1 的宽度优先搜索子术语集扩展对应的搜索图.从 n_1:{ax_1}开始递归地添加与该公理集概念相关的公理集合,找到公理集{ax_2,ax_3},生成节点 n_2:{ax_1,ax_2}和 n_3:{ax_1,ax_3},因这两个节点对应的术语集均不包含互补相关公理,需要进一步扩展.找到公理集{ax_3,ax_4,ax_5},对公理集进行排序.得到公理序列ax_4-ax_5-ax_3,将其顺序应用到节点 n_2 和 n_3,生成节点 n_4:{ax_1,ax_2,ax_4}和 n_5:{ax_1,ax_3,ax_4}.n_4 为互补概念术语集,判断 n_4 对应的术语集关于概念 A_1 的可满足性,因 n_4 对应的术语集{ax_1,ax_2,ax_4}关于 A_1 不可满足,该术语集即为不可满足子术语集 T'.

可以看出,使用如上启发式方式进行搜索一方面可以提高查找 dead 节点的速度,另一方面使得找到的术语集是极小的.换句话说,若算法 3 第 13 行得到的 dead 节点对应的子术语集 T'关于概念 C 不可满足,则 T'是一个 MUPS 且是所有 MUPS 中元素最少的不可满足子术语集.显然,如果宽度优先搜索方法针对搜索图的每一层遍历术语集 T 中所有公理时,该结论成立.下面证明基于概念相关选择函数和互补概念进行公理选择,该结论依然成立.由定义 4 可知,通过概念相关选择函数仅扩展与待扩展元素概念相关的公理集合,对于不相关的公理不予考虑.这是因为概念不相关公理无法直接应用到概念可满足性检测过程中.例如,对于 T_1 中考虑概念 A_1 的可满足性时,第 1 次扩展生成{ax_1},第 2 次扩展时仅需将 ax_2 和 ax_3 加入到术语集中,这是因为只有 A_2 和 A_3 出现在 ax_1: $A_1 \sqsubseteq A_2 \sqcap A_3$ 中.而像 ax_6: $A_6 \sqsubseteq A_1 \sqcap \exists r.(A_3 \sqcap \neg C \sqcap A_5)$ 与 ax_1 不直接相关不需要扩展.因此,使用概念相关选择函数扩展术语集搜索图中的节点只是将不相关的公理从待测术语集中删除.这样能够在保证正确性的前提下减少待测公理个数.另一方面,算法 2 中的启发式优化策略只是对于同一层(属于同一个 $s(T,C,k)$中的元素)扩展的公理进行排序,每次只扩展 $s(T,C,k)$中的一个元素,对应的待测子术语集规模会保持一致.因此,在宽度优先搜索策略下,得到的不可满足子术语集中的元素是最少的,同时也是极小的,亦即它是关于概念 C 的一个 MUPS.

若将例 1 中公理 ax_1 改为如下形式: $A_1 \sqsubseteq A_2 \sqcup A_3$, 则 n_4 对应的节点关于概念 A_1 是可满足的,需要将公理序列中其余公理也加入到待测术语集中,得到节点 n_6:{ax_1,ax_2,ax_3,ax_4,ax_5},判断 n_6 对应的术语集关于 A_1 的可满足性,若 n_6 对应的术语集关于 A_1 仍是可满足的,需要进一步利用概念选择函数扩展术语集以及其关于概念 A_1 的可满足性,直到待测术语集关于 A_1 不可满足或达到 T,否则,n_6 对应的术语集即为不可满足子术语集 T',如图 2 中虚框内所示术语集{ax_1,ax_2,ax_3,ax_4,ax_5}.

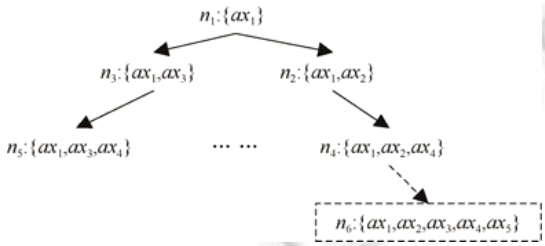

Fig.2 Terminology expansion process based on breadth-first-search w.r.t. concept A_1

图 2 关于概念 A_1 的宽度优先搜索的子术语集扩展过程

3.3 基于深度优先搜索的公理集快速选择

上一节算法的优点在于若 dead 节点对应的术语集关于概念 C 不可满足,则一定能找出 T 中关于 C 的一个 MUPS.但是由于宽度优先搜索需要考虑的节点个数较多,搜索 dead 节点的速度较慢.下面给出一种基于深度优先搜索策略的术语集快速选择算法.

假设一个关于概念 C 不可满足的术语集为 T,基于深度优先搜索的公理集子术语集快速扩展算法同样为术语集 T 构造一个关于 C 的搜索图 $G(V,E,r)$,汇节点 r 对应 $C \sqsubseteq D$ 形式的公理,对于搜索图 G 中的每个节点 $v \in V$,v

都对应术语集 T 的一个子集.算法主要思想如下:通过为输入的术语集 T 构造一个搜索图 $G(V,E,s)$,采用深度优先搜索策略查找图中的dead节点.因为dead节点对应的术语集包含互补相关公理,需要判断其对应的不可满足子术语集 T'是否满足概念 C.若 T'不满足 C,则返回不可满足子术语集为 T',否则,需要回溯到父节点,将当前使用选择函数生成的术语集作为待测术语集进行概念 C 可满足性检测.具体算法如下所示.

算法 4. 基于深度优先搜索的子术语集快速扩展算法.

输入:不可满足概念 C,术语集 T;

输出:一个不可满足子术语集 T'.

```
1.  k←1,T'←∅,W=∅
2.  while W≠T do
3.    S←s(T,C,k)
4.    Σ←s(T,C,k+1)\s(T,C,k)
5.    if W=∅
6.      W←S
7.    select an axiom ax∈Σ
8.    while ax∈W
9.      Σ←Σ\(ax)
10.   if Compl(W∪{ax}) is false
11.     W←W∪{ax}
12.   if Compl(S'∪{ax}) is true
13.     if C is not satisfiable w.r.t. W∪{ax}
14.       T'←W∪{ax}
15.     else
16.       while C is satisfiable w.r.t.s(T,C,k+1)
17.         k←k+1
18.         T'←s(T,C,k)
19.     return T'
20.   k←k+1
21. return T
```

与上一小节介绍的基于宽度优先搜索策略查找 dead 节点的方法相比,算法通过单层循环即可构造术语集 T 关于概念 C 的搜索图 $G(V,E,r)$,从而 dead 节点.同样地,k 从 1 开始递归产生待测术语集加入到 $s(T,C,k)$中.S 为第 k 次使用选择函数获得的公理集合,而Σ为第 k+1 次使用选择函数获得的新的公理集合.初始化待测术语集 W 为 $s(T,C,k)$,对于Σ中的任何一个公理 ax,如果 $W\cup\{ax\}$是互补概念术语集,那么表明该术语集对应的节点为 dead 节点,判断该术语集是否满足概念 C.若 $W\cup\{ax\}$不满足 C,则 $W\cup\{ax\}$作为结果返回,否则,将当前的 $s(T,C,k)$作为子术语集进行概念 C 可满足性检测,这部分与算法 2 一致;$W\cup\{ax\}$中未包含互补相关公理,说明该节点为 live 节点,需要进一步扩展,将 ax 作为元素加入到当前节点所在的术语集中作为下一轮循环中的父节点.

算法使用了如下启发式来优化搜索过程:(1) 第 7 行选择公理 ax 的过程中,对Σ中的公理进行排序,按“包含原子概念否定的公理-与包含原子概念否定的公理概念相关的公理-其他公理”顺序进行排序,可以提高查找 dead 节点的速度;(2) 算法每一次循环时,将本次迭代中用到的公理从待测术语集中删除,减少下次循环时的待测公理个数;(3) 将搜索过的 dead 节点对应的子公式进行标记,当搜索到的 pending 节点与之前已标记的子公式相同时,直接将 pending 节点转换为 dead 节点.

采用基于深度优先搜索方法计算 T_1 关于概念 A_1 的不可满足子术语集对应的搜索图仅生成 3 个节点.从 n_1:$\{ax_1\}$开始递归地添加与该公理集概念相关的公理集$\{ax_2,ax_3\}$.因 ax_2 与 ax_4 概念相关且 ax_4 含有原子概念的

否定,优先选择 ax_2.生成节点 n_2:{ax_1,ax_2},因 n_2 对应的术语集不包含互补相关公理,需要进一步扩展.找到公理集 ax_4,生成节点 n_3:{ax_1,ax_2,ax_4}.n_3 为互补概念术语集,判断 n_3 对应的术语集关于概念 A_1 的可满足性,因 n_3 对应的术语集{ax_1,ax_2,ax_4}关于 A_1 不可满足,该术语集即为不可满足子术语集 T'.可以看出,使用深度优先搜索方法可以减少生成节点的个数.针对 dead 节点对应的术语集关于概念 A_1 可满足的情况,处理方式与算法 2 一致,这里不再赘述.

4 实验结果

本文实验在如下环境中进行:Intel(R) core(TM) i7-6700 3.4GHz CPU;8.0GB RAM,Windows 7 OS,而本体解析与概念的可满足性检测均通过调用 OWLAPI(http://owlapi.sourceforge.net/)实现.

4.1 实例评估

本文实验数据采用的是本体调试中的标准测试集.表 1 给出了这些术语集的相关信息,包括本体的名字、表达能力,本体中公理、类、属性、实例(分别对应描述逻辑中的概念、角色、个体)的个数(*C*/*P*/*I*),不可满足概念的个数以及该本体对应的 MUPS 中元素个数的平均值.

Table 1 The description of ontologies

表 1 待测本体的描述

ID	Ontology	Expressivity	Axioms	*C*/*P*/*I*	UnsConcept	$\overline{MUPS}$
1	Economy	*ACH*(*D*)	2 332	339/46/482	51	3.6
2	MadCow	*ALCHOIN*(*D*)	161	54/16/13	1	4
3	Chemical	*ALCH*(*D*)	192	48/9/0	37	7.6
4	Transportation	*ALCH*(*D*)	2 178	445/89/183	62	4.4
5	Koala	*ALCHOIN*(*D*)	69	21/4/6	3	4
6	Pizza	*SHOIN*(*D*)	916	98/8/5	2	3

下面以 Economy 本体(用 *Te* 表示)中的一个概念 Clover 为例,介绍本文中提到的不可满足子术语集扩展优化方法的实现过程,分别为现有黑盒技术(主要思想是基于概念之间的结构相关性选择公理,ConceptRel)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFSminPath)以及基于深度优先搜索的公理集快速选择方法(DFSQuick).

ConceptRel 与 Comp 方法均通过概念相关选择函数来确定待扩展术语集.对术语集 *Te* 和不可满足概念 Clover,s(*Te*,Clover,1)={axe_1,axe_3},s(*Te*,Clover,2)={axe_1,axe_2,axe_3,axe_4,axe_5,axe_6}.可以看出,使用两次选择函数即可得到不可满足子术语集{axe_1,axe_2,axe_3,axe_4,axe_5,axe_6},其中,

axe_1: Clover⊑ Fodder　　axe_2: PlantAgriculturalProduct⊑ AgriculturalProduct

axe_3: Clover⊑ PlantAgriculturalProduct　　axe_4: AgriculturalProduct≡ disjointWith Fodder

axe_5: AnimalAgriculturalProduct≡ PlantAgriculturalProduct　　axe_6: Fodder⊑ OrganicObject

BFSminPath 方法需要以宽度优先方式扩展节点生成术语集搜索图,扩展过程从 Clover⊑⊤开始,使用与当前节点对应的公理概念相关的公理集合进行扩展.可以看出,与 Clover⊑⊤概念相关的公理为{axe_1,axe_3},该术语集不包括互补相关公理,需要继续扩展.进一步地,需要将{axe_1,axe_3}作为搜索图的一个节点.这是因为本文第 3 节研究术语集扩展方法时默认术语集中只包括一个公理与不可满足概念 *C* 相关,也就是说,对于本例中的两个公理 axe_1 与 axe_3 会整合为一条公理 Clover⊑ Fodder⊓PlantAgriculturalProduct 来表示.因此需要将两条公理都作为待测子术语集加入到术语集搜索图对应的节点中.与 axe_1 概念相关的公理为 axe_4 与 axe_6,与 axe_3 概念相关的公理为 axe_2 与 axe_5.那么得到子术语集{{axe_1,axe_4},{axe_1,axe_6},{axe_3,axe_5},{axe_2,axe_3}},这些公理集合都不包括互补相关公理,需要继续扩展.由{axe_1,axe_3}扩展得到{axe_1,axe_3,axe_5}和{axe_1,axe_2,axe_3},由{axe_1,axe_3}扩展得到{axe_2,axe_3,axe_4},进一步扩展{axe_1,axe_2,axe_3}获得{axe_1,axe_2,axe_3,axe_4}.{axe_1,axe_2,axe_3,axe_4}中包括互补相关公理,需要判断概念可满足性检测.由于 Clover 关于术语集{axe_1,axe_2,axe_3,axe_4}不可满足,该术语集即为 *Te* 关于 Clover 的一个 MUPS.

DFSQuick 方法需要以深度优先方式扩展节点生成术语集搜索图,扩展过程同样从 Clover⊑⊤开始,对与当前节点对应的公理概念相关的术语集进行扩展.但这个过程中每一次选择函数得到的公理集合中仅有一个公理参与扩展,这时扩展的术语集不包含互补相关概念才会进一步回溯扩展其他公理.可以看出,与 Clover⊑⊤概念相关的公理为{axe_1,axe_3},选择 axe_1 进行扩展(选择 axe_3 也可以).由 axe_1 可以扩展 axe_4 与 axe_6,由于 axe_4 中包括否定操作算子($C \not\equiv D$ 等价于 $C \sqsubseteq \neg D$ 且 $D \sqsubseteq \neg C$),选择 axe_4,{axe_1,axe_4}不包括互补相关概念而 axe_4 无法继续扩展,需要回溯将 axe_6 也加到待测术语集中.{axe_1,axe_4,axe_6}不包括互补相关概念且 axe_6 无法继续扩展,需要回溯将 axe_3 也加到待测术语集中.由 axe_3 可以扩展 axe_2 与 axe_5.同样地,可以先选择 axe_2,发现得到的术语集{axe_1,axe_2,axe_3,axe_4,axe_6}包含互补相关公理,需要判断其概念可满足性.由于概念 Clover 关于术语集{axe_1,axe_2,axe_3,axe_4,axe_6}不可满足,该术语集即为不可满足扩展阶段得到的结果.

4.2 术语集扩展优化方法比较

在本节中,我们实现了本文中涉及的关于 MUPS 求解的各类优化算法.在表 1 中的标准本体调试数据集上对这些优化方法进行了测试,并将现有的黑盒技术方法与本文中的各类优化方法进行了比较.表 2 给出了分别使用 ConceptRel、Comp、BFSminPath 以及 DFSQuick 这 4 种优化方法时,不可满足子术语集扩展阶段生成同一个 MUPS 的不可满足子术语集时的推理机调用次数(第 4,6,8,10 列)和所生成的术语集元素个数(第 5,7,9,12 列),其中表格第 2 列给出 MUPS 中的元素个数,第 11 列给出了使用 DFSQuick 方法时的回溯次数.值得注意的是,传统黑盒技术(算法 1)在不可满足子术语集扩展阶段选择的公理集合随机性高,无法给出其具体的推理机调用次数及扩展生成的术语集中元素个数.而且传统黑盒技术在每次加入术语集时都需要进行概念可满足性检测,使其推理机调用次数远远超过其他 4 类方法,这里不作比较.

Table 2 The comparison of teminology expansion optimization algorithms

表 2 子术语集扩展优化算法比较

ID	Ontology	MUPS size	ConceptRel		Comp		BFSminPath		DFSQuick		
			Reasoner	size	Reasoner	size	Reasoner	size	Reasoner	BackTrack	size
1	Economy	3	2	14	1	14	1	**3**	1	1	**3**
2	MadCow	4	3	10	1	10	1	**4**	1	1	**4**
3	Chemical	9	8	83	1	83	1	**9**	1	7	19
4	Transportation	4	3	81	1	81	1	**4**	1	3	7
5	Koala	4	3	11	1	11	1	**4**	1	2	6
6	Pizza	3	2	17	2	17	2	**3**	2	5	7

从表 2 可以看出,本文给出的 3 种优化方法从推理机调用次数和待测术语集规模方面均优于现有的 MUPS 求解方法.

1) 针对标准术语集在不可满足子术语集扩展阶段,本文中的优化方法在大多数情况下仅需调用一次推理机即可.本文中基于互补概念的方法仅在出现互补公理时判断其概念可满足性.本文的本体测试集中公理多以互补概念合取范式形式导致概念可满足.这使得仅调用一次推理机即可得到不可满足子术语集.对于 Pizza 本体,因包含数量限定量词导致的概念不可满足使得 3 种优化方法调用次数与 ConceptRel 方法相同.

2) 在不可满足子术语集扩展阶段得到的术语集元素个数方面,由于 Comp 方法只是考虑使用概念相关选择函数进行扩展,生成公理个数与使用 ConceptRel 方法相同.而在 BFSminPath 以及 DFSQuick 方法中由于优化了每次选择函数扩展时的公理集选择方式,所生成的术语集规模相对较少.这使得在术语集收缩阶段其推理机调用次数也相对减少,可以适当减少求解 MUPS 的执行时间.而 BFSminPath 方法由于选用宽度优先搜索策略,在不可满足子术语集扩展阶段即可得到一个 MUPS 且得到的 MUPS 个数小于等于 MUPS 个数的平均值,省去了不可满足子术语集收缩阶段的工作.这部分也可通过图 3 中的运行时间看出结果.

图 3 给出了分别使用现有黑盒技术(CR)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFS)以及基于深度优先搜索的公理集快速搜索方法(DFS)时,不可满足子术语集扩展阶段运行时间(加后缀_E 表示)和求一个 MUPS 的总体运行时间(加后缀_A 表示).图 3 中横轴表示表 1 中的本体术语集,纵轴表示的是运行时间,单位为毫秒(ms).可以看出,本文给出的 3 种优化方法均优于现有的黑盒技术方法,且

针对规模较大且复杂的本体,其效率更为明显,如 *T*3.

从图 3 可以看出,使用 BFSminPath 与 DFSQuick 时求解 MUPS 的运行时间优于 ConceptRel 与 Comp 方法,而 BFSminPath 与 DFSQuick 针对不同的本体其效率差异并不明显(*T*1 与 *T*2 是 DFSQuick 效率高,其他本体是 BFSminPath 效率高).BFSminPath 的优点在于能够在不可满足子术语集扩展阶段直接得到一个 MUPS,减少了收缩阶段的运行时间,而缺点在于需要考虑术语集中所有公理.DFSQuick 的优点在于每次选择函数生成的公理集合中只需选择一个即可,不需要遍历所有节点,可以快速查找不可满足子术语集.缺点在于如果选择节点不合适,需要多次回溯才能找到不可满足子术语集.分析数据集 *T*1~*T*6 发现,当通过一次概念相关选择函数得到的公理集合中的多个公理属于同一个 MUPS 时,DFSQuick 需要完成多次回溯,降低运行效率.因此,可以通过本体术语集中的公理之间逻辑关系选择优化策略:对于一个不可满足概念 *C*,若使用选择函数得到的子术语集 *T* 中公理之间包含相同概念,则建议使用 BFSminPath;而当 *T* 中公理元素过多时,建议使用 DFSQuick.

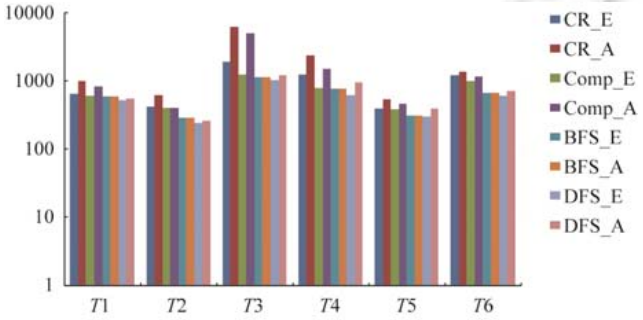

Fig.3 The comparison of runtime for terminology expansion and computing MUPS

图 3 不同优化算法的术语集扩展与 MUPS 求解时间对比

4.3 求解所有 MUPS 优化

前面介绍了求解术语集关于不可满足概念的一个 MUPS 的优化方法.而现有的多数本体测试工具更多地关注于求解一个不可满足概念的所有 MUPS 的方法.如本文第 3.1 节中介绍,现有的求解所有 MUPS 的方法主要是采用碰集树的方法,其优化技术主要在于剪枝操作与节点重用,相关工作已应用到 Pellet 推理机中.本文中的术语集扩展优化方法是针对求解单个 MUPS 的优化方法,可以直接应用于求解所有 MUPS 方法之中,可以有效地提高运行效率.图 4 给出了分别将现有黑盒技术(CR)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFS)以及基于深度优先搜索的公理集快速搜索方法(DFS)应用到求解所有 MUPS 的方法时的运行时间(加后缀_All 表示求解全部 MUPS).图 4 中横轴表示表 1 中的本体术语集,纵轴表示运行时间,单位为毫秒(ms).可以看出,将术语集扩展优化的方法也可以优化求解所有 MUPS 的任务.同样,针对规模较大且复杂的本体,其效率更为明显,如 *T*3 和 *T*4.这是因为针对复杂本体,选择公理的启发式策略更有优势.

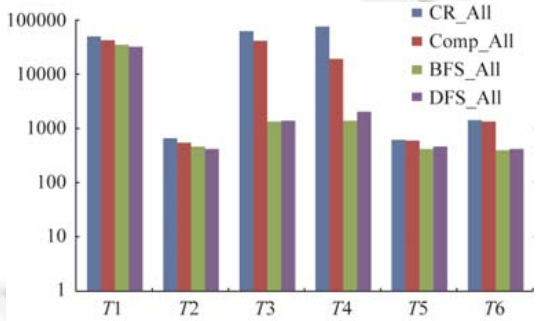

Fig.4 The comparison of runtime for computing all MUPS

图 4 不同优化算法求解所有 MUPS 的时间对比情况

5 总 结

本文结合互补概念和术语集的搜索图提出了一种不可满足子术语集扩展优化方法,构造与术语集扩展对应的搜索图,并分别采用宽度优先搜索和深度优先搜索策略查找不可满足子术语集.基于互补概念的术语集扩展方法仅在待测术语集包含互补概念时才进行概念可满足性检测,大大减少了推理机的调用次数.基于宽度优先搜索的公理集选择方法通过宽度优先搜索方法查找不可满足子术语集扩展对应的搜索图中的 dead 节点,该节点对应的子术语集在大多数情况下是一个极小不可满足子术语集,减少了术语集收缩阶段的操作.而深度优先搜索策略通过启发式方法从选择函数获得的公理集合中优先选择公理进行术语集扩展,大大减少了搜索图中节点的生成个数及待测术语集的规模.实验结果表明,本文给出的术语集扩展优化方法在一定程度上提高了求解 MUPS 的效率.

References:

[1] Staab S, Studer R. Handbook on ontologies. Int'l Journal of Information Management, 2009.

[2] Li P, Jiang YC, Wang J. Modular ontology reuse based on conservative extension theory. Ruan Jian Xue Bao/Journal of Software, 2016,27(11):2777−2795 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4920.htm [doi: 10.13328/j.cnki.jos.004920]

[3] Doms A, Schroeder M. GoPubMed: Exploring PubMed with the gene ontology. Nucleic Acids Research, 2005,33(Suppl. 2): 783−786.

[4] Yang YH, Du JP, Ping Y, Wang J. Ontology-Based intelligent information retrieval system. Ruan Jian Xue Bao/Journal of Software, 2015,26(7):1675−1687 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4622.htm [doi: 10.13328/j.cnki.jos.004622]

[5] Baader F, Calvanese D. The Description Logic Handbook: Theory, Implementation, and Applications. 2nd ed., Cambridge University Press, 2007.

[6] Kalyanpur A, Parsia B, Sirin E, Hendler J. Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics, 2005,3(4):268−293.

[7] Schlobach S, Cornet R. Non-Standard reasoning services for the debugging of description logic terminologies. In: Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence, 2003. 355−362.

[8] Schlobach A, Huang ZS, Cornet R, Harmelen F. Debugging incoherent terminologies. Journal of Automated Reasoning, 2007,39(3): 317−349.

[9] Kalyanpur A, Parsia B, Horridge M, Sirin E. Finding all justifications of OWL DL entailments. In: Proc. of the Semantic Web, the 6th Int'l Semantic Web Conf., the 2nd Asian Semantic Web Conf., ISWC/ASWC. 2007. 267−280.

[10] Baader F, Pealoza R. Automata-Based axiom pinpointing. Journal of Automated Reasoning, 2010,45(2):91−129.

[11] Ye YX, Ouyang DT, Su J. Entailment-Based axiom pinpointing in debugging incoherent terminologies. In: Proc. of the 2nd Int'l Workshop on Semantic Technologies. 2015. 105−115.

[12] Baader F, Suntisrivaraporn B. Debugging SNOMED CT using axiom pinpointing in the description logic EL+. In: Proc. of the 3rd Int'l Conf. on Knowledge Representation in Medicine. 2008.

[13] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987,32:57−95.

[14] Schlobach S. Diagnosing terminologies. In: Proc. of the 20th National Conf. on Artificial Intelligence. 2005. 670−675.

[15] Friedrich G, Shchekotykhin K. A general diagnosis method for ontologies. In: Proc. of the Int'l Semantic Web Conf. 2005. 232−246.

[16] Shchekotykhin K, Friedrich G, Fleiss P, Rodler P. Interactive ontology debugging: Two query strategies for efficient fault localization. Journal of Web Semantics, 2012,12:88−103.

[17] Jannach D, Schmitz T, Shchekotykhin K. Parallel model-based diagnosis on multi-core computers. Journal of Artificial Intelligence Research, 2016,55:835−887.

[18] Parsia B, Sirin E, Kalyanpur A. Debugging OWL ontologies. In: Proc. of the 14th Int'l World Wide Web Conf. 2005. 633−640.

[19] Kalyanpur A, Parsia B, Sirin E. Black box techniques for debugging unsatisfiable concepts. In: Proc. of the 2005 Int'l Workshop on Description Logics. 2005.

[20] Horrocks I, Kutz O, Sattler U. The even more irresistible SROIQ. In: Proc. of the 10th Int'l Conf. on Principles of Knowledge Representation and Reasoning. 2006,6:57–67.

附中文参考文献:

[2] 李璞,蒋运承,王驹.基于保守扩充理论的模块化本体重用.软件学报,2016,27(11):2777–2795. http://www.jos.org.cn/1000-9825/4920.htm [doi: 10.13328/j.cnki.jos.004920]

[4] 杨月华,杜军平,平源.基于本体的智能信息检索系统.软件学报,2015,26(7):1675–1687. http://www.jos.org.cn/1000-9825/4622.htm [doi: 10.13328/j.cnki.jos.004622]

崔仙姬(1986－),女,吉林图们人,博士,讲师,CCF 专业会员,主要研究领域为语义Web,自动推理.

张俊星(1969－),男,博士,教授,主要研究领域为民族信息处理.

何加亮(1977－),男,博士,讲师,主要研究领域为物联网,移动互联网应用.

高健(1983－),男,博士,副教授,CCF 专业会员,主要研究领域为人工智能,约束求解.