基于可满足性计数的($\neq$,=)约束工作流鲁棒性验证

翟治年[1],王 刚[2],郑志军[1],彭艳斌[1],潘志刚[1],王中鹏[1]
(1. 浙江科技学院信息与电子工程学院,浙江杭州 310023;
2. 中国核电工程有限公司河北分公司民用工程研究设计所,河北石家庄 050011)

摘 要: 工作流将业务过程分解为有序的步骤并分配人力资源加以执行.资源分配受访问控制约束及资源异常干扰,存在可满足性和鲁棒性问题.而其鲁棒性验证又依赖于其可满足性判定,目前通过求可满足性的一个解来完成.本文提出另一种途径,通过统计解的个数来完成判定.特别地,通过多项式计数归约为有求解器可用的 # SAT 问题,给出了互斥和绑定约束下的可满足性计数算法.实验表明,相对目前时间复杂度最低的可满足性求解算法,该可满足性计数算法显著提高了实际判定效率和适用规模.

关键词: 工作流;授权;约束;资源分配;可满足性
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2015)11-2298-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.11.024

Verification of ($\neq$, =) Constrained Workflow Robustness Based on Satisfiability Counting

ZHAI Zhi-nian[1], WANG Gang[2], ZHENG Zhi-jun[1], PENG Yan-bing[1], PAN Zhi-gang[1], WANG Zhong-peng[1]
(1. *School of Information and Electronics Engineering, Zhejiang University of Science and Technology, Hangzhou, Zhejiang* 310023, *China*;
2. *Civil Engineering Research & Design Institute, China Nuclear Power Engineering Limited Corporation Hebei Branch, Shijiazhuang, Hebei* 050011, *China*)

Abstract: In a workflow system, a business process is decomposed into orderly steps, and then human resources are allocated to execute them. The process of resource allocation is constrained by access control and influenced by resource exceptions, so there exist the issues of satisfiability and robustness. Especially, the verification of its robustness relies on the judgment of its satisfiability, which is currently realized via finding one of the satisfiability solutions. This paper proposes another approach, in which the total number of all the solutions is counted to make a satisfiability judgment. Especially, the satisfiability counting under exclusion ($\neq$) and binding (=) constraints is reduced to the problem of # SAT with solvers available by a polynomial-time counting reduction. Experiments show that, compared to the existing lowest-time satisfiability solving algorithm, our satisfiability counting algorithm achieves remarkable improvement in judgment efficiency and applicable scale.

Key words: workflow; authorization; constraint; resource allocation; satisfiability

1 引言

工作流将业务过程分解为有序的步骤,统一分配人力资源予以执行.执行期资源分配受到系统访问控制(Access Control, AC)策略(主要包括授权和约束)的限制,存在可行与否的问题——称为工作流可满足性(Workflow Satisfiability, WS)[1].WS 是 AC 策略规划的基本目标.由于资源本身难免出现各种异常,还应通过合理的 AC 规划,使资源分配过程具备一定程度的鲁棒性——称为工作流 k 鲁棒性(Workflow k Robustness, $WR^{(k)}$),或工作流 k 弹性[1].

WS 要求不违反授权和约束的资源分配是存在的.它关系到工作流能否实施,反映了 AC 策略是否正确.WS 验证在 AC 规划阶段完成,由于规划试错往往引起反复验证,其执行效率非常重要.仅在职责分离约束下的 WS($\neq$)已经是 NP 完全的[1].不过,WS 的指数阶求解研究始终在持续[2~6].1999 年,Bertino 通过穷举搜索来枚举 WS 的所有解[2],为了支持复杂约束,通过逻辑程

收稿日期:2014-10-13;修回日期:2015-07-06;责任编辑:覃怀银
基金项目:国家自然科学基金(No.51376162);浙江省自然科学基金(No.LY13F050005)

序规则来描述 WS 问题,而其解的验证通过稳定模型语义来实现,对互斥和绑定约束下的 WS 即 WS(≠,=)),其时间复杂度不低于 $O(|U|^{|S|+1}|S|^2)$(S、U 分别为步骤和用户集).随后,Atluri 将其扩展为允许委托授权[3],但核心算法不变.2008 年,Crampton 在委托背景下给出了一种穷举搜索的 WS 求解算法[4],其解的验证以朴素方式进行,对 WS(≠,=),时间复杂度降低至 $O(|U|^{|S|}|S|^2)$,2010 年,Wang 等又进一步指出,授权用户数量超过 $|S|$ 的步骤必可获得无冲突的资源分配,在 WS 求解时可以忽略.特别是 2013 年,Crampton 通过归约为集合最大权划分问题[7],将 WS(≠)求解时间降至 $O(2^{|S|}(|C|+|U|^2))=O(2^{|S|}|U|^2)$($C$ 为约束集),是目前最好的理论结果[6],但其空间占用为 $O^*(2^{|S|})$,限制了它在普通机器配置上的求解规模.这些研究主要集中于 WS 决策(记为 * WS),只求找到任意一个解,未涉及 WS 计数(记为 # WS),不能给出所有解的数量.而一个经 * WS 验证可行的工作流未必能持续可靠运行,授权用户出现离职、病休等异常,可能导致现有解失效,工作流不再可行.

$WR^{(k)}$正是对资源异常时工作流鲁棒性的度量.它要求任意删除 k 个用户后 WS 均成立[1].$k=0$ 时,$WR^{(k)}$ = WS.然而 $k>0$ 时,$WR^{(k)}$相当于 $C(|U|,k)$个 WS,其验证开销极为可观.Wang 等已证明$WR^{(k)}$是 NP 难的[1].目前只能借助 WS 决策甚至枚举进行 WS 判定,进而完成$WR^{(k)}$验证,算法的实际性能不够理想.

本文将互斥和绑定约束下的 $WR^{(k)}$验证方法进行研究,主要贡献是:(1)建立了一条基于 WS 计数而非决策的 $WR^{(k)}$验证途径.借助 # SAT 求解器进行 # WS(≠)求解,其 WS(≠)判定性能优于目前时间复杂度最低的 * WS(≠)算法,有利于提高 $WR^{(k)}$(≠)的验证效率.(2)给出了一种从 # WS(≠,=)到 # WS(≠)的多项式时间归约算法,结合(1)的 # WS(≠)算法,其整体性能优于将 # WS(≠,=)直接归约为 # SAT 求解,有利于提高 $WR^{(k)}$(≠,=)的验证效率.

2　问题定义

本节给出可满足性与 k 鲁棒性相关定义,约定 S 表示步骤集,U 表示用户集.

定义 1　工作流$(S,\leqslant)$是步骤的偏序集,工作流的每次执行称为一个案例.

定义 2　(1)AC 策略是一个三元组$\langle A,X,B\rangle$,其中:

(1)$A\subseteq S\times U$ 为授权,表示以 S 为步骤集的工作流在用户集 U 上的执行资格分配.若$\langle s,u\rangle\in A$,称 u 是 s 的授权用户,s 是 u 的授权步骤.$U_A(s)$表示 s 的授权用户集.

(2)$X\subseteq S\times S$ 为互斥约束,表示步骤间的职责分离关系.若$(s,s')\in X$,则在同一案例中,s 和 s' 不能由同一用户执行.互斥禁止某些利益冲突的步骤由同一用户执行,以防止借机欺诈.例如按照财务管理制度,出款和记账必须由不同的用户负责.

(3)$B\subseteq S\times S$ 为绑定约束,表示步骤间的职责绑定关系.若$(s,s')\in B$,则在同一案例中,s 和 s' 必须由同一用户执行.绑定要求某些相关步骤由同一用户执行,以保证业务规则或工作效率等.例如材料采购流程,验货必须由采购申请人来完成.

定义 3　从 S 到 U 的资源分配 π 是 $S\to U$ 函数,它从 U 中为 S 中每个步骤指派唯一的执行用户.工作流的每个案例都要进行相应的资源分配.

定义 4　互斥和绑定约束下的 WS 问题 WS(≠,=)是一个五元组$\langle S,U,A,X,B\rangle$,或者有默认的 S 和 U 时,用一个 AC 策略$\langle A,X,B\rangle$来表示.它的解是从 S 到 U 的资源分配 π,且其满足如下条件:

(1)基于 A:任取 $s\in S$,$\langle s,\pi(s)\rangle\in A$.

(2)不违反 X:任取$(s,s')\in X$,$\pi(s)\neq\pi(s')$.

(3)不违反 B:任取$(s,s')\in B$,$\pi(s)=\pi(s')$.

为强调只需求一个解,可记为 * WS(≠,=),若需要统计解的个数,则记为 # WS(≠,=),而解的个数记为 #$\langle A,X,B\rangle$.

定义 5　互斥和绑定约束下的工作流 k 鲁棒性问题$WR^{(k)}$(≠,=)$(0\leqslant k<|U|)$是一个五元组$\langle S,U,A,X,B\rangle$.有默认的 S 和 U 时,用 AC 策略$\langle A,X,B\rangle$来表示.它要求回答:从 U 中任意删除 k 个用户的集合 U_k 以及相应的授权,所得$\langle S,U-U_k,A-(S\times U_k),X,B\rangle\in$WS(≠,=)是否恒有解.

3　问题求解

为了求解$\langle S,U,A,X,B\rangle\in WR^{(k)}$(≠,=),需判定删除任意 $U_k\subseteq U$ 所得 WS(≠,=)是否成立.本文将通过统计所有解的个数,而不是寻找一个具体的解来完成这一判定.只需对 # WS(≠,=)的一般形式建立求解方法.若绑定约束 B 非空,则$\langle S,U,A,X,B\rangle\in$ # WS(≠,=)的规模可以约简.如下算法 1 在排除一些原本无解的情况后,可将 # WS(≠,=)归约为等价的 # WS(≠).

算法 1　从 # WS(≠,=)到 # WS(≠)的归约

输入:S,U,A,X,B

输出:S',U',A',X',使得 #$\langle S',U',A',X',\varnothing\rangle$ = #$\langle S,U,A,X,B\rangle$,返回 NULL 表示 #$\langle S,U,A,X,B\rangle=0$

1. $S'\leftarrow\varnothing$;
2. $A'\leftarrow\varnothing$;
3. $X'\leftarrow\varnothing$;

4. for ($s\in S$) $U_A(s)\leftarrow\varnothing$;
5. for ($\langle s,u\rangle\in A$) $U_A(s)\leftarrow U_A(s)\cup\{u\}$;//求所有 $U_A(s)$
6. 求 $G_B=\langle S,B\rangle$的每个连通片 C 的顶点集 S_C; //S_C 中的所有步骤必须由同一用户执行
7. for ($S_C\in\{S_C\}$)
8. for ($s\in S_C$)
9. $C(s)\leftarrow C$; //设置 $C(s)$的值,它表示步骤 s 在 G_B 中所属的连通片,相当于 s 的一个属性
10. end for
11. end for
12. for ($S_C\in\{S_C\}$)
13. $U_{S_C}\leftarrow\bigcap_{s\in S_C}U_A(s)$; //$U_{S_C}$是 S_C 中所有步骤的公共授权用户集
14. if ($U_{S_C}=\varnothing$) return NULL; //根据授权限制,没有用户可同时执行 S_C 中所有步骤,原问题无解
15. $S'\leftarrow S'\cup\{s_C\}$; //$s_C$ 从 S_C 中任意选取,此后便固定下来,作为 S_C 的代表元,取所有代表元的集合为 S'
16. for ($u\in U_{S_C}$) $A'\leftarrow A'\cup\{\langle s_C,u\rangle\}$; //$S_C$ 的每个公共授权用户被授权执行 S_C 的代表步骤
17. end for
18. for ($(s_i,s_j)\in X$)
19. if ($C(s_i)=C(s_j)$) return NULL; //绑定约束连通片中存在互斥步骤,原问题无解
20. $X'\leftarrow X'\cup\{(s_{C(s_i)},s_{C(s_j)})\}$; //保留两个连通片之间的互斥约束关系
21. end for
22. $U'\leftarrow\varnothing$;
23. for ($\langle s,u\rangle\in A'$) $U'\leftarrow U'\cup\{u\}$;
24. return S',U',A',X'

算法 1 所需时间为 $O(|S|\cdot(|S|+|U|\cdot\log|U|))$. 如下定理 1 表明了算法的正确性.

定理 1 算法 1 返回 NULL 时,$\langle S,U,A,X,B\rangle\in$ WS($\neq$,=)无解. 否则,所得$\langle S',U,A',X',\varnothing\rangle\in$ WS($\neq$)具有与原问题一一对应的解,且 $S'\subseteq S,U'\subseteq U,A'\subseteq A,|X'|\leqslant|X|$.

证明 S 中任何步骤 s 都有其授权用户集 $U_A(s)$(由第 5 步计算),s 只能分配给 $U_A(s)$中的用户执行. 对绑定约束图 G_B 的任何连通片 C,其顶点集 S_C 中的所有步骤必须由同一用户执行. 因此为保证$\langle A,X,B\rangle\in$ WS($\neq$,=)有解,$\bigcap_{s\in S_C}U_A(s)$不能为空(在第 14 步进行检查). 同时,在 S_C 中的步骤之间,显然也不能存在互斥约束(在第 19 步进行检查). 排除这些无解情形后,算法 1 设置的 S'、U'、A'和 X',将使得两个 WS($\neq$,=)问题$\langle S',U',A',X',\varnothing\rangle$与$\langle S,U,A,X,B\rangle$具有一一对应的解,从而相应的计数问题同解. 证明如下:

由第 15 步,$S'\subseteq S$,且 G_B 的每个连通片 C 中,只有唯一的步骤进入 S',记为 s_C,即 $S_C\cap S'=\{s_C\}$.

在第 16 步,$u\in U_{S_C}=\bigcap_{s\in S_C}U_A(s)\subseteq U_A(s_C)=\{u\in U|\langle s_C,u\rangle\in A\}$,故$\langle s_C,u\rangle\in A$,因此 $A'\subseteq A$.

在第 20 步,将$(s_{C(s_i)},s_{C(s_j)})$加入 X'是因为$(s_i,s_j)\in X$并且 s_i,s_j 跨连通片,而 X'中的元组均由此而来,故 $|X'|\leqslant|X|$.

由第 23 步,$U'\subseteq U$.

设$\langle S',U',A',X',\varnothing\rangle\in$WS($\neq$)和$\langle S,U,A,X,B\rangle\in$WS($\neq$,=)的解集分别为 Π'和Π,构造从 Π'到Π 的函数φ,证其为一一对应. 任取 Π'中的资源分配 $\pi':S'\rightarrow U'$,先构造 $\pi:S\rightarrow U$ 如下:任取 $s\in S$,必存在 G_B 的连通片 C 使得$s\in S_C$,而 S_C 的代表元$s_C\in S'$,令 $\pi(s)=\pi'(s_C)$. 我们根据定义 4 证明 $\pi\in\Pi$,

(1)π 是基于 A 的. 任取 $s\in S$,设其在 G_B 中属于连通片 C,若 $s=s_C$,由于 $\pi'\in\Pi'$,π' 基于 A',于是$\langle s_C,\pi'(s_C)\rangle\in A'\subseteq A$,又由 π 的构造有 $\pi(s)=\pi'(s_C)$,故$\langle s,\pi(s)\rangle\in A$. 否则,即 $s\neq s_C$,由第 16 步的设置可知,对进入 A'的每个$\langle s_C,u\rangle$,均有 $u\in U_{S_C}=\bigcap_{s'\in S_C}U_A(s')\subseteq U_A(s)=\{u\in U|\langle s,u\rangle\in A\}$,从而$\langle s,u\rangle\in A$,那么由$\langle s_C,\pi'(s_C)\rangle\in A'$可知$\langle s,\pi'(s_C)\rangle\in A$,即$\langle s,\pi(s)\rangle\in A$,得证.

(2)π 不违反 X. 第 19 步检查绑定连通片内部有互斥约束的无解情形,通过检查后,X 中的约束都是跨连通片的. 此时任取$(s_i,s_j)\in X$,$C(s_i)\neq C(s_j)$,这两个连通片进入 S' 的代表元为 $s_{C(s_i)}$,$s_{C(s_j)}$. 由第 20 步可知$(s_{C(s_i)},s_{C(s_j)})\in X'$,由 $\pi'\in\Pi'$,π' 不违反 X',故 $\pi'(s_{C(s_i)})\neq\pi'(s_{C(s_j)})$,即 $\pi(s_i)\neq\pi(s_j)$,得证.

(3)π 不违反 B. 任取$(s_i,s_j)\in B$,必有 $C(s_i)=C(s_j)$,将此连通片记为 C,由 π 的构造可知,$\pi(s_i)=\pi(s_j)=\pi'(s_C)$,得证.

于是 π 构成$\langle S,U,A,X,B\rangle\in$WS($\neq$,=)的解.

任取 $\pi'\in\Pi'$,按前述方式构造 π,令 $\varphi(\pi')=\pi$,则 φ 是从Π'到Π 的函数. 下面证明 φ 是一一对应:

(1)满射性. 对 Π 中的任何资源分配π,构造 $\pi':S'\rightarrow U'$,任取 $s\in S'$,$\pi'(s)=\pi(s)$. S'中的每个步骤 s 都是来自某个连通片 C. 由于 π 是$\langle S,U,A,X,B\rangle\in$WS($\neq$,=)的解,由算法第 14 步可知 $\bigcap_{s\in S_C}U_A(s)\neq\varnothing$,且任取 $s\in S_C$,$\pi(s)\in\bigcap_{s\in S_C}U_A(s)$,否则 π 为 S_C 中不同步骤指派的用户就不可能相同. 又由算法对 A'的设置,任取 $u\in\bigcap_{s\in S_C}U_A(s)$,$\langle s,u\rangle\in A'$,因此$\langle s,\pi'(s)\rangle=\langle s,\pi(s)\rangle\in A'$,即 π'是基于 A'的. 又由于 π 不违反 X,只要两个连通片 C 和 C'之间存在互斥约束,即$(s_C,s_{C'})\in X'$,π 为 C 和 C'中的步骤指派的用户便不同,特别地,$\pi(s_C)\neq\pi(s_{C'})$,从而 $\pi'(s_C)\neq\pi'(s_{C'})$;由 C 和 C'的任意性,π'不违反 X'. 故 π'是$\langle S',U',A',X',\varnothing\rangle\in$WS($\neq$)的一个解,即 $\pi'\in\Pi'$. 由 φ 的构造易知$\varphi(\pi')=\pi$,得证.

(2)单射性.用反证法.假设存在 $\pi', \pi'' \in \Pi', \pi' \neq \pi''$,使得 $\varphi(\pi') = \varphi(\pi'')$.那么由 $\pi' \neq \pi''$必存在$s \in S'$使得 $\pi'(s) \neq \pi''(s)$.由 φ 的定义,$\varphi(\pi')$和 $\varphi(\pi'')$也将为 s 指派不同的用户,这与 $\varphi(\pi') = \varphi(\pi'')$矛盾,得证.

于是 φ 为一一对应,从而$\langle S', U', A', X', \varnothing\rangle \in$ # WS($\neq$)和$\langle S, U, A, X, B\rangle \in$ # WS($\neq$,=)同解.

算法 1 消去所有的绑定约束,且步骤、用户、授权以及互斥约束的数量也会相应有所减少.对 WS 这样的困难问题,先以多项式时间的归约代价使其规模尽可能降低,对整体性能是有利的,后文将通过实验来验证这一点.

接下来需要给出 # WS($\neq$)的求解方法,本文将其归约为合取范式可满足性计数问题 # SAT 进行求解. # SAT 定义为某个变量集(设其大小为 n)上的一个子句集(设其大小为 m).变量表示原子命题,只能取 1 或 0 值.变量或其否定式称为文字.一组具有析取关系的文字构成一个子句,而子句之间具有合取关系. # SAT 要求统计子句集所有成真赋值的个数,也称模型计数.若将每个授权定义为一个变量 p_{ij},其值为 1/0 表示 s_i是/否指派给 u_j执行,则下面的算法 2 可将 # WS($\neq$)归约为 # SAT.

算法 2　从 # WS($\neq$)到 # SAT 的归约

输入:S, U, A, X

输出:子句集 F

1. 用与算法 1 同样的方法求所有 $U_A(s)$;
2. $F \leftarrow \varnothing$;
3. for ($s_i \in S$)
4. $F \leftarrow F \cup \{p_{ij} \mid u_j \in U_A(s_i)\}$;//生成子句保证每个 s_i都由其授权用户执行
5. for ($U_A(s_i)$中每一对用户 u_j, u_k)
6. $F \leftarrow F \cup \{\{\neg s_{ij}, \neg s_{ik}\}\}$;//生成子句保证 s_i不能同时分配给两个授权用户
7. end for
8. end for
9. for ($(s_i, s_j) \in X$)
10. for ($u_j \in U_A(s_i) \cap U_A(s_j)$) //当互斥步骤没有公共授权用户时,不需要生成约束子句
11. $F \leftarrow F \cup \{\{\neg p_{ik}, \neg p_{jk}\}\}$;//生成子句保证互斥步骤不能同时分配给一个用户
12. end for
13. end for
14. return F

算法 2 所需时间为 $O(|S| \cdot |U| \cdot (|S| \cdot \log|U| + |U|))$.如下定理 2 表明了算法的正确性.

定理 2　算法 2 返回的子句集 F,其成真赋值的数目等于$\langle S, U, A, X\rangle \in$ WS($\neq$)解的数目.

证明　任取$\langle S, U, A, X\rangle \in$ WS($\neq$)的一个解 π,由其可以唯一构造 F 的赋值 α_π: $\alpha_\pi(p_{ij})$ = true 当且仅当 $\pi(s_i) = u_j$.任取步骤 s_i,由于它一定被指派给 $U_A(s_i)$中的某个用户,第 4 步加入 F 的子句在 α_π 下为真.由于 s_i不会被指派给 $U_A(s_i)$中两个不同的用户,第 6 步加入的子句在 α_π 下为真.任取步骤 s_i, s_j,若$(s_i, s_j) \in X$,则它们不会被指派给同一用户,第 11 步加入的子句在 α_π 下为真.因此 α_π 是 F 的成真赋值.下面证明 F 也只有这样的成真赋值.

用反证法.假设存在一个成真赋值 α,对$\langle S, U, A, X\rangle \in$WS($\neq$)的任意解 π,均有 $\alpha \neq \alpha_\pi$.根据 α 构造唯一的二元关系 π_α:$\langle s_i, u_j\rangle \in \pi_\alpha$ 当且仅当 $\alpha(p_{ij})$ = true.由算法 2 可知,对 F 中出现的所有原子命题 p_{ij},均存在 s_i, u_j使得$\langle s_i, u_j\rangle \in A$.任取步骤 s_i,根据第 4 和 6 两步加入的子句可知,仅在 $U_A(s_i)$中有唯一的 u_j 使得$\alpha(p_{ij})$ = true.因此 π_α 是从 S 到 U 的函数,即资源分配,且是基于 A 的.任取步骤 s_i, s_j,若$(s_i, s_j) \in X$,根据第 11 步加入的子句可知,不存在 u_k使得$\alpha(p_{ik}) = \alpha(p_{jk})$ = true,因此 $\pi_\alpha(s_i) \neq \pi_\alpha(s_j)$.因此 π_α 也不违反 X,从而 π_α 是$\langle S, U, A, X\rangle \in$WS($\neq$)的一个解.然而,$\alpha_{\pi_\alpha}(p_{ij})$ = true 当且仅当 $\pi_\alpha(s_i) = u_j$ 当且仅当$\alpha(p_{ij})$ = true,可知 $\alpha = \alpha_{\pi_\alpha}$,这与反证假设矛盾,目标得证.

进一步可知,F 成真赋值的数目等于$\langle S, U, A, X\rangle \in$WS($\neq$)解的数目.

基于以上准备,分别给出 # WS($\neq$)和 # WS($\neq$,=)算法如下.

算法 3　# WS($\neq$)求解

输入:S, U, A, X

输出:# $\langle S, U, A, X\rangle$

1. 以 S, U, A, X 调用算法 2 得到 F;
2. 以 F 及其原子命题集调用 # SAT 求解器,返回求解结果.

算法 4　# WS($\neq$,=)求解

输入:S, U, A, X, B

输出:# $\langle S, U, A, X, B\rangle$

1. 以 S, U, A, X, B 调用算法 1 得到 S', U', A', X';
2. 以 S', U', A', X'调用算法 3,返回求解结果.

二者的时间复杂度依赖于 # SAT 求解器,本文选择 sharpSAT[8,9],这是一个基于 BCP(Boolean Constraint Propogation)和组件缓存思想的求解器,其时间复杂度为 $O(\text{Poly}(n,m)2^n) = O^*(\text{Poly}(m)2^n)$,若不使用隐式 BCP、缓存压缩与清理等可配置特性,则为 $O(n^2 m 2^n) = O^*(\text{Poly}(m)2^n)$.其空间复杂度同样为 $O^*(\text{Poly}(m)2^n)$,主要是组件缓存开销.由于算法 2 产生的子句集包含 $O(|S| \cdot |U|)$个变量,$O(|S| + |S| \cdot |U|^2 + |X|)$个

子句,算法 3 和 4 的时间复杂度将不低于 $O((2^{|U|})^{|S|}|S|^2\cdot|U|^2(|S|+|S|\cdot|U|^2+|X|))$,均弱于相应的穷举搜索算法(分别为 $O(|U|^{|S|}|X|)$和 $O(|U|^{|S|}|X\cup B|)$),算法 3 显著弱于前述文献[6]的算法.算法 3 和 4 的空间复杂度也达到 $O^*(2^{|S|\cdot|U|}\mathrm{Poly}(|X|))$.然而,sharpSAT 有着良好的实测性能[8].将其用于 # WS 求解时,输入规模也只是原始输入的多项式函数.特别地,# WS 生成的子句中绝大部分是二元子句(至多 $|S|$ 个多元子句),而且其中有很多变量多次出现,非常有利于 BCP 发挥作用.由于冲突赋值不产生需要缓存的组件,再加上压缩和清理技术的使用,组件缓存实际占用的空间通常也不会太大.因此,可预期本文算法将有较好的实际性能和一定求解规模.

4 实验研究

本节通过实验来证明本文算法的优势,并与文献[4]的 WS 版本和[6]的 * WS 算法比较.所有算法均以 C++ 实现,实验环境为:2.1GHz Core i7 CPU、8GB RAM、Windows 8.1、Win32.

实验 1 比较算法 3、文献[6]和[4]三者的 WS($\neq$)判定性能,利用随机生成数据进行测试.数据生成规则为:步骤数 5~15、用户比例 0.5~1、约束密度 0.05.在所得 15 组数据上的实验结果如表 1 所示(—表示执行时间超过 1 小时,× 表示内存分配失败).

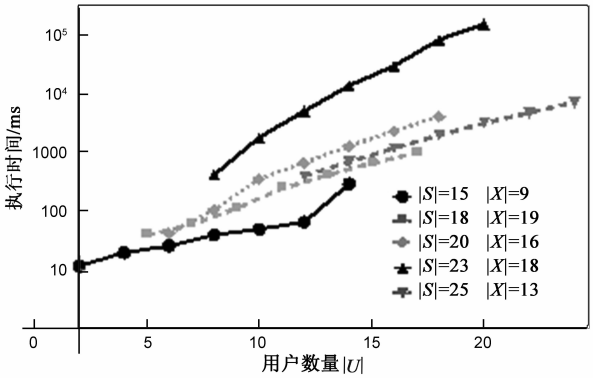
在 14 组数据上,算法 3 较文献[6]降低了时间开销,其中 11 组数据降低了 97% 以上.文献[6]因空间限制不能求解 $|S|=15$、$|U|=10$ 规模的数据(即使 $|X|=4$),但算法 3 的求解能力仍有余地.补充测试表明,当 $|S|=15$、$|U|=11$ 时,即使 $|X|=0$,文献[6]仍不能求解.随后的实验 2 将表明,算法 3 至少可求解 $|S|=25$、$|U|=14$、$|X|=22$ 或 $|S|=25$、$|U|=13$、$|X|=24$ 规模的数据.在输入规模最小的第 1 组数据上,算法 3 的时间开销比文献[6]增大了 450%,但绝对执行时间之差不超过 10ms.在 13 组数据上,算法 3 较文献[4]降低了时间开销,其中 12 组数据降低了 96% 以上.特别是规模大于 $|S|=13$、$|U|=8$ 的 4 组数据,文献[4]的求解时间均超过 1h,而算法 3 所需时间不超过 60ms.在输入规模最小的前 2 组数据上,算法 3 的时间开销比文献[4]增大了 900% 以上,但绝对执行时间之差不超过 11ms.

实验 2 进一步研究算法 3 执行时间与输入规模之间的关系.从表 1 可以看出,随着 $|S|$ 的增长,执行时间整体上呈增长趋势,但是由于 $|X|$ 或 $|U|$ 的不同,在某些局部也有起伏.下面将给定 $|S|$,考查随 $|X|$ 或 $|U|$ 的变化,执行时间如何变化.图 1(a)是对不同的 $|S|$ 分别生成一组约束,然后逐渐增大 $|U|$ 的值,测试执行时间的变化,由图可见,执行时间必然随 $|U|$ 的增大而增长.图 1(b)是对不同的 $|S|$ 和 $|U|$ 分别生成一组约束,然后逐渐从中随机删减约束,测试执行时间的变化.由图可见,执行时间随约束减少呈降低趋势.某些局部有约束减少而执行时间增大的情况,这反映了问题结构变动的影响.一个可能的原因是,被删减约束生成的子句,扩大了 BCP 的作用范围,因此不作删减反而更有利于整体求解.

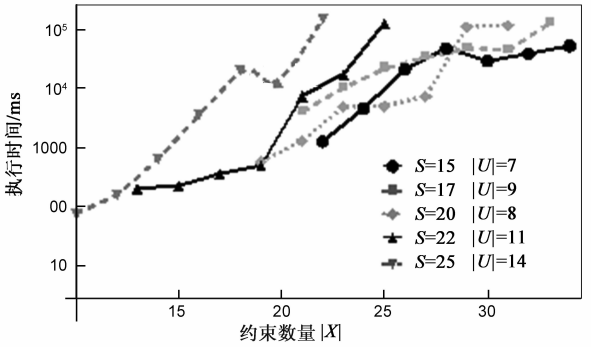
表 1 文献[6]、[4]与算法 3 的 WS($\neq$)判定时间(ms)对比

	$\|S\|$	$\|U\|$	$\|X\|$	文献[4]	文献[6]	算法3	比[4]降低	比[6]降低
1	5	3	1	<1	2	11	>-1000%	-450%
2	7	3	2	1	25	10	-900%	60%
3	8	6	2	296	149	13	96%	91%
4	9	4	3	54	142	11	80%	92%
5	9	7	3	4925	489	16	99.7%	97%
6	9	8	2	14974	968	13	99.9%	98%
7	10	5	3	1725	629	13	99%	98%
8	10	8	3	166630	1837	17	99.99%	99%
9	10	9	3	496899	2834	21	99.996%	99%
10	11	5	3	7727	2122	13	99.8%	99%
11	12	5	2	55566	7835	15	99.97%	99.8%
12	13	8	5	—	36325	25	∞	99.9%
13	14	9	5	—	173079	20	∞	99.99%
14	14	9	6	—	169434	23	∞	99.99%
15	15	10	4	—	×	54	∞	∞

实验 3 # WS($\neq$, =)可直接归约为 # SAT,也可经由 # WS($\neq$)归约为 # SAT 求解(算法 4).为了比较两条途径的性能,并验证算法 4 求解一定规模数据的能力,随机生成数据进行测试,规则为:步骤数 15~30,用户比例 0.5~1,约束密度 0.1,其中绑定约束比例为 0.3,并滤掉存在约束冲突的情形.在所得 30 组数据上的实验结果如表 2 所示.在 70% 的数据上,算法 4 较直接归约降低了总体时间开销.在其中 24% 的数据上,算法 4 可以在 100s 内给出结果,而直接归约无法计算.对其余 76% 的数据,时间开销平均降低了 47%.可见通过两步归约,问题规模得到约简,有效提高了总体求解性能.在 20% 的数据上,算法 4 所需时间超过直接归约,其中 83% 的数据时间开销平均增大 17%,有 16% 数据增大了 498%.测试表明,约简过程所需时间相对总体时间开销微不足道.规模约简反而导致时间开销增大的现象,主要是由于问题结构改变引起的.当一个绑定约束连通片被约简为一个代表元时,该连通片对外的所有互斥约束都被集中到这一代表元.在互斥约束总



(*a*) 执行时间与用户数量的关系



(*b*) 执行时间与约束数量的关系

图1　算法3执行时间与输入规模的关系

量不变的前提下,若局部耦合显著增强,则可能不利于进一步分解问题,导致求解效率降低.

表 2　直接归约与算法 4 的 # WS(≠,=)求解时间(ms)对比

	|S|	|U|	|X|	直接	算法 4	降低
1	15	9	16	41	42	2%
2	15	13	8	37	37	0
3	15	13	14	80	73	9%
4	16	12	12	73	84	-15%
5	16	13	12	40	39	3%
6	19	10	16	412	62	85%
7	19	12	20	99	102	-3%
8	20	11	22	117	58	50%
9	20	13	17	135	122	10%
10	21	10	21	10135	516	95%
11	21	11	31	1019	665	35%
12	21	14	23	1286	1310	-2%
13	21	15	19	10271	2204	79%
14	21	21	26	—	11040	∞
15	23	11	29	1714	796	54%
16	23	15	26	—	90526	∞
17	23	19	17	113	165	-46%
18	23	19	17	2227	13317	-498%
19	23	23	27	2095	2527	-21%
20	24	17	35	—	12210	∞
21	24	20	27	—	—	0
22	24	22	17	372	282	24%
23	24	22	27	88804	52993	40%
24	25	13	35	88000	37469	57%
25	27	21	44	—	99333	∞
26	27	23	26	18977	6736	65%
27	29	20	35	101302	50695	50%
28	29	27	31	—	—	0
29	29	28	17	212	19	91%
30	29	28	33	—	9302	∞

5　结论与下一步工作

本文通过归约为 # SAT 给出了 # WS(≠,=)和 # WS(≠)算法,以解决相应 $WR^{(k)}$ 验证中的 WS 判定问题,并证明了两个归约算法的正确性.基于 sharpSAT 求解器的实验表明,与当前时间复杂度最低的 * WS(≠)算法比较,本文算法的 WS 判定效率有大幅度提高,求解规模有明显提高.与穷举搜索方式的 # WS(≠)算法比较,本文算法的优势更为显著.进一步的实验表明,较之于直接归约为 # SAT 进行 # WS(≠,=)求解,先归约为 # WS(≠)可明显提高时间性能.利用 # WS 计数结果中的信息来简化 $WR^{(k)}$ 验证,将是我们下一步工作的重点.

参考文献

[1] WANG Q, Li N. Satisfiability and resiliency in workflow authorization systems[J]. ACM Transactions on Information and System Security, 2010, 13(4): 1 - 35.

[2] BERTINO E, FERRARI E, ATLURI V. The specification and enforcement of authorization constraints in workflow management systems[J]. ACM Transactions on Information System Security. 1999, 2(1): 65 - 104.

[3] ATLURI V, BERTINO E, FERRARI E, et al. Supporting delegation in secure workflow management systems[A]. Proceedings of the 17th Annual Conference on Data and Application of Security[C]. Berlin, GER: Springer, 2003. 190 - 202.

[4] CRAMPTON J, KHAMBHAMMETTU H. Delegation and satisfiability in workflow systems[A]. Proceedings of the 13th ACM Symposium on Access Control Models and Technologies [C]. NY, USA: ACM, 2008. 31 - 40.

[5] CRAMPTON J, GUTIN G. Constraint expressions and workflow satisfiability[A]. Proceedings of the 18th ACM Sympo-

sium on Access Control Models and Technologies[C]. NY, USA: ACM, 2013. 73 – 84.
[6] CRAMPTON J, GUTIN G, YEO A. On the parameterized complexity and kernelization of the workflow satisfiability problem [J]. ACM Transactions on Information and System Security, 2013, 16(1): 1 – 31.
[7] BJORCLUND A, HUSFELDT T, KOIVISTO M. Set partitioning via inclusion-exclusion[J]. SIAM Journal on Computing, 2009, 39(2): 546 – 563.
[8] THURLEY M. Sharp SAT counting models with advanced caching and implicit BCP[A]. Proc of the 9th International Conference on Theory and Applications of Satisfiability Testing [C]. Berlin, GER: Springer, 2006. 424 – 429.
[9] THURLEY M. SharpSAT[EB/OL]. https://github.com/marcthurley/sharpSAT, 2013 – 02.

作者简介

翟治年 男, 1977 年 5 月生, 河北张家口人. 2012 年毕业于华南理工大学计算机应用技术专业, 获工学博士学位. 现为浙江科技学院信息与电子工程学院讲师, 主要从事信息安全与访问控制方面的研究.
E-mail: zhaizhinian@gmail.com

王中鹏(通信作者) 男, 1966 年 3 月出生, 辽宁省锦州市人. 2004 年毕业于北京邮电大学信号与信息处理专业, 获工学博士, 现为浙江科技学院信息与电子工程学院副教授, 主要从事通信信号处理和网络信息安全的研究工作.
E-main: wzp1966@sohu.com