软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(7):1538−1550 [doi: 10.3724/SP.J.1001.2011.03859]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

求解 QBF 问题的启发式调查传播算法*

殷明浩[1,3], 周俊萍[1,2+], 孙吉贵[2,3], 谷文祥[1]

[1](东北师范大学 计算机学院,吉林 长春 130117)

[2](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[3](吉林大学 教育部符号计算与知识工程重点实验室,吉林 长春 130012)

Heuristic Survey Propagation Algorithm for Solving QBF Problem

YIN Ming-Hao[1,3], ZHOU Jun-Ping[1,2+], SUN Ji-Gui[2,3], GU Wen-Xiang[1]

[1](College of Computer Science, Northeast Normal University, Changchun 130117, China)

[2](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[3](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012, China)

+ Corresponding author: E-mail: zhoujp877@nenu.edu.cn

Yin MH, Zhou JP, Sun JG, Gu WX. Heuristic survey propagation algorithm for solving QBF problem. *Journal of Software*, 2011,22(7):1538−1550. http://www.jos.org.cn/1000-9825/3859.htm

Abstract: This paper presents a heuristic survey propagation algorithm for solving Quantified Boolean Formulae (QBF) problem. A QBF solver based on the algorithm is designed, namely HSPQBF (heuristic survey propagation algorithm for solving QBF). This solver is a QBF reasoning engine that incorporates Survey Propagation method for problem solving. Using the information obtained from the survey propagation procedure, HSPQBF can select a branch accurately. Furthermore, when handling the branches, HSPQBF uses efficient technology to solve QBF problems, such as unit propagation, conflict driven learning, and satisfiability directed at implication and learning. The experimental results also show that HSPQBF can solve both random and QBF benchmark problems efficiently, which validates the effect of using survey propagation in a QBF solving process.

Key words: artificial intelligence; quantified Boolean formulae problem; QBF(quantified Boolean formulae) solver; factor graph; survey propagation; conflict driven learning; satisfiability directed implication and learning

摘 要: 提出了一种启发式调查传播算法,并基于该算法设计了一种 QBF(quantified Boolean formulae)求解器——HSPQBF(heuristic survey propagation algorithm for solving QBF)系统.它将 Survey Propagation 信息传递方法应用到 QBF 求解问题中.利用 Survey Propagation 作为启发式引导 DPLL(Davis,Putnam,Logemann and Loveland)算法,选择合适的变量进行分支,从而可以减小搜索空间,并减少算法回退的次数.在分支处理过程中,HSPQBF 系统结合了单元传播、冲突学习和满足蕴涵学习等一些优秀的 QBF 求解技术,从而能够提高 QBF 问题的求解效率.实验结果表明,HSPQBF 无论在随机问题上还是在 QBF 标准测试问题上都有很好的表现,验证了调查传播技术在 QBF 问题求解中的实际价值.

* 基金项目: 国家自然科学基金(60773097, 60803102)
收稿时间: 2008-07-18; 定稿时间: 2010-03-29

关键词: 人工智能;QBF 问题;QBF 问题求解器;因子图;调查传播;冲突学习;满足蕴涵学习
中图法分类号: TP181 文献标识码: A

量化布尔公式(quantified Boolean formulae,简称 QBF)是指带有存在和全称量词前缀的命题逻辑公式.当 QBF 公式的量词中只含有存在量词时,QBF 问题就转化为命题可满足(SAT)问题.因此,QBF 问题通常可以看作是 SAT 问题的泛化.Garey 等人证明 QBF 问题的计算复杂性高于 SAT 问题,是 PSPACE(polynomial space)完备的[1],它可以在多项式时间内与一致性规划、验证、非单调推理、知识推理等问题进行相互转换[2−4].由于其重要的理论价值和应用价值,QBF 问题的研究受到了广泛的重视.

Gent 等人设计了第一款不完备的 QBF 求解器 WalkQSAT,该求解器在启发式设计中利用了 SAT 问题求解器 Walksat 的启发式方法[5].Rowley 等人设计了 QBF 求解器 WATCHEDCSBJ,它扩展了 SAT 求解器 ZChaff 中的监视文字这种放松的数据结构[6].Giunchiglia 等人设计了 QBF 求解器 QuBE,该求解器使用了子句学习技术,它采用的数据结构与 WATCHEDCSBJ 一样,通过使用监视文字这种放松的数据结构而得到的[7].Rintanen 等人设计了 QSAT 求解器,它使用了 MOMS 启发式方法[8].Zhang 等人设计了 Quaffle 求解器,该求解器采用了变量状态独立衰退总和的启发式方法,并把子句学习技术融入到求解器中[9].

事实上,几乎所有高效的 QBF 问题求解器都是基于 DPLL(Davis,Putnam,Logemann and Loveland)算法设计的,影响 DPLL 算法效率的最主要的因素之一是分支的选取.为此,研究人员采用了各种方法来确定 DPLL 算法的选择分支.Cadoli 等人设计的 Evaluate 求解器采用了一种最朴素的方法——随机选取方法,该系统也是最早将 SAT 问题的求解方法引入到 QBF 的求解方法的系统之一[10].Zhang 等人设计的 Quaffle 求解器借鉴了 Zchaff 求解器中使用的变量状态独立下降的启发式方法[9].Rowley 等人设计的 WALKQSAT 求解器借助的是 Walksat 启发式方法:通过 Walksat 求解器找到一个文字来确定分支[2].Giunchiglia 等人设计的 QuBE 求解器采用的是 bohm 和 jw2 的启发式方法:根据在最短的子句中出现次数最多的文字来确定分支[7].

近年来,命题可满足问题(SAT)的求解能力有了很大程度的提高,但依然存在一些开放性问题.1999 年,Zecchina 等人在《Nature》杂志上发表的论文指出,在求解随机 K-SAT 问题时,存在着从可满足到不可满足的相变,当$\alpha_{cluster}<\alpha<\alpha_c$($\alpha_c\approx4.27$,$\alpha_c$ 为子句个数与变量个数的比值)时,SAT 问题会出现求解困难的区域[11].2002 年,Mezard 等人在《Science》杂志上给出了调查传播(survey propagation,简称 SP)算法,该算法源于统计物理学,可以在稍高于线性时间内判断具有 1 000 000 变量规模的处于难解区域的 SAT 问题[12].

从本质上说,SP 算法可以看作是一种基于 DPLL 的 SAT 回退搜索算法.它首先计算一组满足变量赋值的概率边缘分布,并逐步确定具有最大分布值的变量.令人惊奇的是,SP 算法几乎很少回退,换言之,以 SP 启发式来引导搜索在大多数情况下都是高效的.本文的主要工作之一即是将 SP 方法应用于求解 QBF 问题的分支选择上,利用 SP 方法作为启发式引导搜索来减少搜索过程中的回退次数,从而加速求解过程.

与目前大部分 QBF 求解器相同,HSPQBF(heuristic survey propagation algorithm for solving QBF)系统也是一款基于 DPLL 算法的 QBF 求解器.该系统综合利用多种方法来提高 QBF 问题的求解效率.在分支选择过程中,HSPQBF 使用 SP 方法提供的全局信息确定分支,以缩小搜索空间,从而减少算法回退次数.同时,分支处理过程中 HSPQBF 利用冲突推理、冲突学习、满足蕴涵学习等技术来减小搜索空间,加速了问题求解.

1 相关概念

为了讨论方便,首先介绍本文需要的相关概念.

定义 1(量化布尔公式). 设 PS 是一个有限的命题符号集合,PS 上的量化布尔公式集合 $QPROPps$ 是如下递归定义的最小符号集合:

(1) T,⊥,v 属于集合 $QPROPps$(其中,T 表示真,⊥表示假,$v\in PS$ 且 v 是变量);

(2) 若ϕ,φ属于集合 $QPROPps$,则$\neg(\phi)$,$(\phi\wedge\varphi)$,$(\phi\vee\varphi)$,$(\phi\Rightarrow\varphi)$,$(\phi\Leftrightarrow\varphi)$,$(\phi\oplus\varphi)$属于集合 $QPROPps$;

(3) 若ϕ属于集合 $QPROPps$ 且 x 属于 PS,则$\forall x(\phi)$,$\exists x(\phi)$属于集合 $QPROPps$.

由定义可以看出,一个量化布尔公式是一个抽象的命题符号集合,没有什么意义,更无真假可言.但如果对 PS 中的每个命题符号给以真或假的解释,那么量化布尔公式 F 就变成一个有真假意义的命题了.将这个解释记为 I,公式 F 在解释 I 下的真值记为 $F(I)$,递归的计算方法如下:

(1) 若 $F=\forall x(\phi)$,则 $F(I)=\min(\{\phi_{x\leftarrow 0}(I),\phi_{x\leftarrow 1}(I)\})$;

(2) 若 $F=\exists x(\phi)$,则 $F(I)=\max(\{\phi_{x\leftarrow 0}(I),\phi_{x\leftarrow 1}(I)\})$.

一般地,量化布尔公式 F 可以表示为 $Q_1x_1\ldots Q_nx_n\phi$,其中,$Q_i\in\{\forall,\exists\}$,$i=1,\ldots,n$,$\phi$既可以用合取范式表示,也可以用析取范式表示.文中$\phi$采用合取范式,$Var(\phi)=\{x_1,x_2,\ldots,x_n\}$,对于任意的布尔变量 $x_i(i=1,\ldots,n)$,有且仅有一个量词对它进行约束.由全称量词约束的变量称为全称变量,由存在量词约束的变量称为存在变量.

定义 2(量化布尔公式的可满足性问题). 量化布尔公式的可满足性问题是指:给定一个量化布尔公式,判断是否存在一个解释,使得该量化布尔公式在该解释下为真.如果存在该公式的一个解释使得该公式为真,则该公式是可满足的(SAT);否则,是不可满足的(UNSAT).

需要注意的是,对于一个 QBF 公式 $Q_1x_1\ldots Q_nx_n\phi$,如果相邻的量词是不同的,则其量词顺序不可改变.例如,公式$\forall x_1\exists x_2(x_1\vee x_2)\wedge(\neg x_1\vee\neg x_2)$和公式$\exists x_2\forall x_1(x_1\vee x_2)\wedge(\neg x_1\vee\neg x_2)$是逻辑不等价的;但是如果相邻的量词是相同的,则可以交换量词的顺序.例如,$\exists x_1\exists x_2\phi\equiv\exists x_2\exists x_1\phi$,$\forall x_1\forall x_2\phi\equiv\forall x_2\forall x_1\phi$.因此,通常将相邻的相同量词合并,这样,公式可表示为

$$Q_1X_1\ldots Q_kX_k\phi \tag{1}$$

其中,$X_i(i=1,\ldots,k)$是具有相同量词的变量集.由于相邻的相同量词都在一组,这时,公式(1)中的全称量词和存在量词是交替出现的,每个量词约束一个变量集,文中采用如公式(1)所示的 QBF 公式.如果公式(1)的量词个数为 k,称该公式为 kQBF 公式.同时,将公式(1)中最外层的量词层定义为第 1 层,依次类推,最内层的量词层为第 k 层.由于当最内层为存在量词且 k 为奇数时,问题是 Σ_k^P-完全的;当最内层为存在量词且 k 为偶数时,问题是 Π_k^P-完全的;当最内层为全称量词且 k 为奇数时,问题是 Π_k^P-完全的;当最内层为全称量词且 k 为偶数时,问题是 Σ_k^P-完全的.它们都是多项式级别的,因而,本文只研究最内层是存在量词的 QBF 问题.

2 HSPQBF 算法框架

DPLL 算法是求解 SAT 问题的最为高效的算法之一,目前几乎所有的 QBF 求解器都是基于 DPLL 算法设计的,图 1 给出了求解 QBF 问题的 DPLL 算法的基本框架.

在 DPLL 算法中,$C_\varnothing$表示空子句,$C_{all\forall}$表示由全称文字组成的子句,l_v表示变量 v 所对应的文字,$l_\exists$表示存在文字集合,$l_\forall$表示全称文字集合,$Q.E|_{v=\text{true}}$ 表示将变量 v 赋值为真后得到的公式集合,$Q.E|_{v=\text{false}}$ 表示将变量 v 赋值为假后得到的公式集合.在预处理阶段(preprocess),DPLL 算法首先使用纯文字规则、单元文字规则等推理规则简化 QBF 公式,从而判断算法是否满足终止条件:如果化简后得到的公式集合为空,则原 QBF 公式可满足;否则,若化简后得到的公式集合包含空子句或者存在一个仅由全称文字组成的子句,则原 QBF 公式不可满足.若经过预处理阶段无法判断该 QBF 问题是否可满足,则选择被 QBF 公式的最外层量词层约束的一个变量拆分与或树,得到两个公式集合 $Q.E|_{v=\text{true}}$ 和 $Q.E|_{v=\text{false}}$.若该变量对应的文字是存在文字,只要其中的一个公式集合是可满足的,就可以保证原公式集合可满足;若该变量对应的文字是全称文字,那么原公式集合可满足当且仅当 $Q.E|_{v=\text{true}}$ 和 $Q.E|_{v=\text{false}}$ 均为可满足的.由 DPLL 算法可以看出,选择一个合适变量来拆分与或树是影响整个算法效率的最关键因素之一.

我们在基本的 DPLL 算法的基础上设计了 HSPQBF 算法,算法的整体框架如图 2 所示.算法中记号的含义与 DPLL 中的相同,$C_\varnothing$表示空子句,$C_{all\forall}$表示由全称文字组成的子句,l_v表示变量 v 所对应的文字,$l_\exists$表示存在文字集合,$l_\forall$表示全称文字集合,$Q.E|_{v=\text{true}}$ 表示将变量 v 赋值为真后得到的公式集合,$Q.E|_{v=\text{false}}$ 表示将变量 v 赋值为假后得到的公式集合.在预处理阶段,HSPQBF 算法使用单文字规则对 QBF 公式进行化简.在分支选择阶段,HSPQBF 算法采用 SP 方法作为启发式(SP_choosevariable),该方法通过提供全局信息确定分支,缩小了搜索空间,从而减少了算法的回退次数,这部分内容将在第 3 节中给出详细介绍.在分支处理阶段,首先利用冲突推理

规则对 QBF 公式进行推导,该推理有 3 种返回值,分别为 conflict,satisfaction 和 undetermined.如果 E 在当前赋值下为假,即返回值为 conflict,则进行冲突分析;如果 E 在当前赋值下为真,即返回值为 satisfaction,则进行满足分析;如果 E 在当前的赋值下不能判断其真值,返回值是 undetermined,继续分支选择.在这个阶段,利用冲突学习、满足蕴涵学习等技术减小搜索空间,加速了问题求解,第 4 节将主要介绍这些方法.

```
Procedure DPLL(Q.E)
1.  preprocess(Q.E);
2.  if E=∅ then return SAT;
3.  if (C∅∈E)∨(Call∀∈E) then return UNSAT;
4.  v←choosevariable(Q.E);
5.  if (lv∈l∃)
6.  then return DPLL(Q.E|v=true) or DPLL(Q.E|v=false)
7.  if (lv∈l∀)
8.  then return DPLL(Q.E|v=true}) and DPLL(Q.E|v=false)
```

Fig.1 DPLL algorithm

图 1 DPLL 算法

```
Procedure HSPQBF(Q.E)
1.  preprocess(Q.E);
2.  if (E=∅) then return SAT;
3.  if (C∅∈E)∨(Call∀∈E) then return UNSAT;
4.  result=deduce();
5.  if (result=conflict)
6.  then analyze_conflict();
7.  if (result=satisfaction)
8.  then analyze_satisfaction();
9.  v←SP_choosevariable(Q.E);
10. if (lv∈l∃)
11. then return HSPQBF(Q.E|v=true) or HSPQBF(Q.E|v=false)
12. if (lv∈l∀)
13. then return HSPQBF(Q.E|v=true) and HSPQBF(Q.E|v=false)
```

Fig.2 HSPQBF algorithm

图 2 HSPQBF 算法

3 基于 Survey Propagation 方法的分支选择

SP 算法可以看作是一种基于 DPLL 的 SAT 回退搜索算法,它首先计算一组满足变量赋值的概率边缘分布,并逐步确定具有最大分布值的变量.令人惊奇的是,SP 算法几乎很少回退,换言之,SP 算法在启发式引导中大多数情况下都是高效的.SP 算法在计算边缘分布的过程中,采用一种消息传递技术,并利用调查迭代函数反复地计算传递的消息.一旦传递的消息收敛,就利用启发式消解过程找到发生“冻结”的变量,并确定这些“冻结”的变量逐步简化布尔公式.本节将讨论如何利用 SP 方法在 QBF 求解过程中进行分支选择.

3.1 利用Survey Propagation进行分支选择

在求解 QBF 问题时,使用 SP 方法进行分支选择.在计算边缘分布的过程中,SP 方法在因子图上传递消息,它通过计算边缘分布从全局的角度给出相应的信息,使 QBF 求解器可以更准确地选取分支,缩小搜索空间,减少算法回退的次数,提高算法的效率.

3.1.1 因子图(factor graph)

因子图是一个无向图 $G=(V,E)$,其中,V 为节点集合,E 为弧集合.因子图包含两类节点:变量节点(图中用圆圈表示)和子句节点(图中用方块表示).若变量 i 出现在子句 a 中,就用一条弧连接变量节点 i 与子句节点 a.根据变量 i 在子句中出现的形式,因子图包含两类弧:若变量 i 以正的形式出现在子句 a 中,则称该弧为正弧,用一条实线连接节点 i 与节点 a;若变量 i 以负的形式出现在子句 a 中,则称该弧为负弧,用一条虚线连接节点 i 与节点 a.对每一个变量节点 i,用 $V(i)$表示与变量节点 i 相连的所有子句节点的集合,用 $V_+(i)$表示与文字 i 相连的所有子句节点的集合,用 $V_-(i)$表示与文字$\neg i$ 相连的所有子句节点的集合.$V(i)\backslash b$ 表示除去子句节点 b 的所有与变量 i 相连的子句节点的集合.类似地,对每一个子句节点 a,用 $V(a)=V_+(a)\cup V_-(a)$表示邻接变量节点的集合.例如,对于 QBF 公式$\forall x_1\exists x_2(x_1\vee x_2)\wedge(\neg x_1\vee\neg x_2)$,可以用因子图(如图 3 所示)表示它的命题公式部分.其中,a 子句表示 $x_1\vee x_2$,b 子句表示$\neg x_1\vee\neg x_2$.

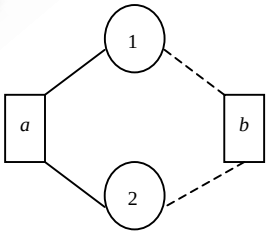


Fig.3 An simple example of a factor graph

图 3 一个简单的因子图示例

3.1.2 消息传递

SP 算法中传递的消息分为两种:一种是子句向变量传递的消息,另一种是变量向子句传递的消息.消息的更新是利用调查迭代函数来实现的.

(1) 子句 a 向变量 i 传递消息的迭代函数:子句向变量传递的消息称为调查,用 $\eta_{a\to i}$ 表示(其中,$\eta_{a\to i}\in[0,1]$).

$$\eta_{a\to i}=\prod_{j\in V(a)\backslash i}\left[\frac{\Pi^{u}_{j\to a}}{\Pi^{u}_{j\to a}+\Pi^{s}_{j\to a}+\Pi^{0}_{j\to a}}\right] \tag{2}$$

其中,若 $V(a)\backslash i$ 为空,则 $\eta_{a\to i}=1$.

(2) 变量 i 向子句 a 传递消息的迭代函数:变量向子句传递的消息是一个三元组 $\Pi_{i\to a}=(\Pi^{u}_{i\to a},\Pi^{s}_{i\to a},\Pi^{0}_{i\to a})$.它们的迭代函数分别是

$$\Pi^{u}_{i\to a}=\left[1-\prod_{b\in V^{u}_{a}(i)}(1-\eta_{b\to i})\right]\prod_{b\in V^{s}_{a}(i)}(1-\eta_{b\to i}) \tag{3}$$

$$\Pi^{s}_{i\to a}=\left[1-\prod_{b\in V^{s}_{a}(i)}(1-\eta_{b\to i})\right]\prod_{b\in V^{u}_{a}(i)}(1-\eta_{b\to i}) \tag{4}$$

$$\Pi^{0}_{i\to a}=\prod_{b\in V(i)\backslash a}(1-\eta_{b\to i}) \tag{5}$$

其中:$V^{u}_{a}(i)$ 表示与 i 连接的所有使变量 i 不满足子句 a 的子句节点的集合(除了节点 a 以外).如果 $V^{u}_{a}(i)$ 是空集,那么相应的乘积是 1;$V^{s}_{a}(i)$ 表示与 i 连接的所有使变量 i 满足子句 a 的子句节点的集合(除了节点 a 以外).如果 $V^{s}_{a}(i)$ 是空集,那么相应的乘积是 1.

图 4 给出上述消息传递函数的直观解释.它包含两种消息:一种是子句向变量发送的消息,表示变量的赋值满足子句的概率.例如图 4 中左图所示,子句 a 包含 3 个变量 x,y,z,假设子句 a 获得来自变量 y 和 z 的消息,知道变量 y 和 z 不能满足子句 a,那么,要想满足子句 a,就必须向变量 x 发送调查($\eta_{a\to x}>0$);另一种是变量向子句发送的消息,表示变量的赋值不能满足子句的概率.例如图 4 中右图所示,5 个子句 a,b,c,d,e 都包含变量 x,其中,子句 a,b,c,e 包含文字 x,子句 d 包含文字 $\neg x$.子句 a,b,c,e 向变量 x 发送调查,因为变量 x 以否定的形式出现在子句 d 中,根据这些调查,那么,变量 x 的赋值不能满足子句 d.

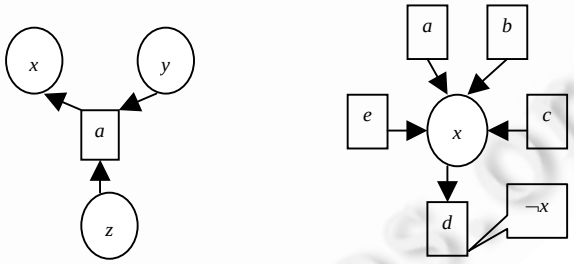


Fig.4 Two types of messages passed

图 4 两种消息传递类型

3.1.3 基于 Survey Propagation 的分支选择

首先给出 SP 算法,如图 5 所示.SP 算法的输入参数为最大迭代次数 $t_{\max}$ 和要求精度 ε.算法首先在 $t=0$ 时随机初始化所有调查,然后反复地利用调查迭代函数(Update_SP())依次更新调查(第 3 行),当连续两次迭代得到的所有调查差的绝对值小于要求精度时,表明算法收敛,得到所有的调查.SP 算法得到的调查可用于估计每个变量在所有解中的统计特性(称为偏量),这种偏量分为正偏量和负偏量.正偏量是指,当随机选取一个簇时,变量 i 被约束("冻结")到 $x_i=1$ 的概率.类似地,负偏量是指,当随机选取一个簇时,变量 i 被约束("冻结")到 $x_i=0$ 的概率.下面给出正偏量和负偏量的计算公式.

正偏量:

$$W_i^{(+)}=\frac{\hat{\Pi}_i^{+}}{\hat{\Pi}_i^{+}+\hat{\Pi}_i^{-}+\hat{\Pi}_i^{0}} \tag{6}$$

负偏量:

$$W_i^{(-)}=\frac{\hat{\Pi}_i^{+}}{\hat{\Pi}_i^{+}+\hat{\Pi}_i^{-}+\hat{\Pi}_i^{0}} \tag{7}$$

其中:

$$\hat{\Pi}_i^{+}=\left[1-\prod_{a\in V_+(i)}(1-\eta_{a\to i}^{*})\right]\prod_{a\in V_-(i)}(1-\eta_{a\to i}^{*}) \tag{8}$$

$$\hat{\Pi}_i^{-}=\left[1-\prod_{a\in V_-(i)}(1-\eta_{a\to i}^{*})\right]\prod_{a\in V_+(i)}(1-\eta_{a\to i}^{*}) \tag{9}$$

$$\hat{\Pi}_i^{0}=\prod_{a\in V(i)}(1-\eta_{a\to i}^{*}) \tag{10}$$

根据变量的正偏量和负偏量,可以得到变量的偏量值为

$$W_i=\max(W_i^{(+)},W_i^{(-)}) \tag{11}$$

在求解 QBF 公式时,SP 方法可以为求解提供全局信息,该信息可用于估计每个变量在所有解中的统计特性,进而指导 DPLL 过程的分支选择.在选择分支的过程中,首先根据 QBF 公式的量词层按照从小到大的顺序选择变量.对于同一量词层的变量,通过调用 SP 过程可以得到变量的正偏量 $W_i^{(+)}$ 和负偏量 $W_i^{(-)}$.根据偏量值,可以将变量分为 3 类:偏向变量($W_i^{(+)}$~1 或 $W_i^{(-)}$~1)、平衡变量($W_i^{(+)}\sim W_i^{(-)}$ 且 $W_i^{(0)}$ 很小)、约束不足变量($W_i^{(0)}$~1).由于选择平衡变量会改变问题的解簇的结构,损失大量解簇;约束不足变量不能提供足够的求解信息,因此选择已经“冻结”的偏向变量作为待分支的变量,即那些具有最大$|W_i^{(+)}-W_i^{(-)}|$值的变量.根据约束变量的量词的不同,对变量采用的赋值方式也不同.如果约束当前分支变量的量词是存在量词,首先考察将变量赋值为其偏向值的分支.这是因为,对于存在变量,只要与或树的一个分支是可满足的就可以保证原公式可满足,因此,应该选择有较多满足解的子空间;如果约束当前分支变量的量词是全称量词,因为这时只有当该变量对应节点的两个分支都是可满足的才可以保证原公式可满足,所以首先考察将变量赋值为其偏向值的反的分支;如果该分支不可满足,则不需要继续考察另一个分支.实验结果表明,上述方法可以显著提高 DPLL 方法的求解效率.

Procedure *Survey Propagation*($t_{\max}$,ε)
1. **if** (t=0) **then** *initialize_*$\eta_{a\to i}$;
2. **for** t=1 **to** t=$t_{\max}$
3. $\eta_{a\to i}(t)$=*Update_SP*();
4. **if** ($|\eta_{a\to i}(t)-\eta_{a\to i}(t-1)|_{all}<\varepsilon$)
5. **then Goto** 7
6. **endfor**
7. **if** (t=$t_{\max}$) **then return** UN-CONVERGED;
8. **else return** $\eta_{a\to i}^{*}=\eta_{a\to i}(t)$.

Fig.5 Survey Propagation algorithm

图 5 Survey Propagation 算法

4 分支处理

在选择好待处理的 QBF 子公式后,进入分支处理阶段.分支处理的目的在于判断搜索树当前分支所对应的 QBF 子公式的可满足性.在这一阶段采用了冲突推理、冲突学习和满足蕴涵学习等技术对搜索空间进行剪枝,以提高分支处理的效率.

4.1 冲突推理

在判断给定 QBF 公式的可满足性之前,先利用冲突推理技术对该公式进行化简.这一过程主要采取单文字规则和冲突规则.

定义 3(单元子句). 给定一个 QBF 公式 F,C 为 F 中的子句,称 C 是一个单元子句当且仅当:(1) C 中有且仅有一个存在文字 l;(2) C 中所有全称文字的量词层都大于 l 的量词层.其中,称存在文字 l 为单元文字.

定义 4(单文字规则). 如果 QBF 公式 F 中存在单元文字 l,则将 l 赋值为真,得到一个新公式 F'.

给定 QBF 公式 F,C 为单元子句,若 l 为 C 中的单元文字,显然,在所有使 C 为真的赋值中,l 的真值都应该为真,因此可以得到如下结论:

命题 5. 给定 QBF 公式 F,设 F'为在 F 上应用单文字规则后得到的新的 QBF 公式,则 F 可满足当且仅当 F'可满足.

定义 6(赋值冲突). 给定一个 QBF 公式 F,C 为 F 中的子句,如果在赋值过程中 C 满足:(1) C 中所有存在文字的赋值都为假;(2) C 中所有全称文字的赋值都不为真.这时,称 F 出现赋值冲突,称 C 为冲突子句.

定义 7(冲突规则). 给定一个 QBF 公式 F,在赋值过程中,如果当前分支存在冲突子句,则在该分支中 F 不可满足.

在应用单文字规则和冲突规则后,需要根据推理的结果作进一步的处理.当推理的结果是 conflict 时,表示当前分支不可满足,需要进行冲突学习,具体内容在第 4.2 节中加以讨论;当推理结果为 satisfaction 时,表示当前分支可以满足,需要进行满足蕴涵学习,具体内容在第 4.3 节中加以讨论.

4.2 冲突学习

如果当前 DPLL 分支出现冲突,就表明当前分支下的子 QBF 公式是不可满足的.这时可以进行冲突学习,冲突学习的主要目的是记录当前存在的冲突分支.这样,当随后的搜索过程中遇到同样的情况时,就可以不用继续搜索而直接判断当前的搜索分支是不可满足的.在阐述这部分内容之前,先给出该部分所涉及的定义以及命题.

定义 8(决策层). 变量 a 的决策层是 a 在与或树中所对应节点的层数,这里规定与或树根节点的层数是 1.

定义 9(先行子句). 变量 a 的先行子句是应用单文字规则确定 a 真值的子句.

定义 10(归结子句). 设子句 C_1 包含文字$\{l_1,l_2,\ldots,l_m,a\}$,子句 C_2 包含文字$\{l_{m+1},l_{m+2},\ldots,l_{m+n},\neg a\}$,则子句 C_1 与 C_2 关于变量 a 的归结子句包含文字$\{l_1,l_2,\ldots,l_m,l_{m+1},l_{m+2},\ldots,l_{m+n}\}$.

命题 11. 给定 QBF 公式 F,设 F'是在公式 F 中添加 F 的归结子句后得到的,则 F 与 F'的满足性相同.

冲突学习主要是由冲突分析过程实现的,图 6 给出了该过程的算法框架.该过程首先判断当前 QBF 公式是否存在冲突子句(*find_conflicting_ clause*()),如果存在,则利用 *generate_clause*()过程生成新的学习子句(*new_cl*).将生成的新子句加入到数据库中(*add_clause*()),这实际上就是冲突学习过程.然后判断新子句的决策层是否为 0,如果是,则算法结束;否则,回退到新子句对应的决策层.这样, 在下次搜索到同样的分支时无需继续向下搜索,从而缩小了搜索空间.

```
Routine analyze_conflict()
1.  conflictcl=find_conflict_clause();
2.  newcl=generate_clause(conflict_cl);
3.  add_clause(new_cl);
4.  if (dl(new_cl)≠0)
5.  then backtrack(dl(new_cl));
6.  end
```

Fig.6 Analyze conflict routine

图 6 冲突分析子过程

4.3 满足蕴涵学习

如果当前 DPLL 分支为可满足,则进行满足蕴涵学习.满足蕴涵学习的主要目的在于记录当前存在的满足

分支.这样,当在随后的搜索过程中遇到同样的情况时,也可以不用继续搜索而直接判断当前的搜索分支是可满足的,进而直接搜索全称变量的另一个分支.在说明满足蕴涵学习之前,首先给出相关的定义和命题.

定义 12(扩展的合取范式). 给定一个 CNF 范式ϕ,扩展的合取范式$\phi'=\phi\vee EN_1\vee EN_2\vee\ldots\vee EN_m$,其中,$EN_1$,$EN_2$,…,$EN_m$表示短语(即文字的合取).

命题 13. 若命题公式ϕ分别用 CNF 和 DNF 表示并且其真值相同,即$\phi=C_1\wedge C_2\wedge\ldots\wedge C_n=EN_1\vee EN_2\vee\ldots\vee EN_t$,则量化布尔公式 $F=Q_1X_1\ldots Q_kX_k\phi=Q_1X_1\ldots Q_kX_k(C_1\wedge C_2\wedge\ldots\wedge C_n)$等价于 $F'=Q_1X_1\ldots Q_kX_k\phi'=Q_1X_1\ldots Q_kX_k(C_1\wedge C_2\wedge\ldots\wedge C_n\vee EN_1'\vee EN_2'\vee\ldots\vee EN_m')$,其中,$C_i(1\leqslant i\leqslant n)$表示子句,$EN_i(1\leqslant i\leqslant t)$和 EN_i' $(1\leqslant i\leqslant m)$表示短语.

定义 14(单元短语). 给定一个以扩展合取范式表示的 QBF 公式 $F=Q_1X_1\ldots Q_kX_k\phi'=Q_1X_1\ldots Q_kX_k(\phi\vee EN_1\vee EN_2\vee\ldots\vee EN_m)$,设 $EN_i(1\leqslant i\leqslant m)$为 F 中的短语,称 EN_i 是一个单元短语当且仅当:(1) EN_i 中有且仅有一个全称文字 l;(2) EN_i 中所有存在文字的量词层都大于 l 的量词层.

定义 15(蕴涵规则). 如果以扩展合取范式表示的 QBF 公式 F 中存在单元短语 EN,则将仅有的全称文字 l 赋值为假,得到一个新公式 F'.

给定以扩展合取范式表示的 QBF 公式 F,EN 为单元短语,若 l 为 EN 中仅有的全称文字,显然,在所有使 C 为真的赋值中,l 的真值都应该为真,因此可以得到如下结论:

命题 16. 给定 QBF 公式 F,设 F'为在 F 上应用单文字规则得到的新的 QBF 公式,则 F 可满足当且仅当 F' 可满足.

满足蕴涵学习是由满足分析子过程实现的,图 7 给出了满足分析子过程的基本框架.与冲突学习存储的单元子句不同,满足蕴涵学习存储的是单元短语.满足分析子过程首先判断当前公式是否存在满足短语(*find_sat_entity*()),如果存在,则子过程执行第 4 行;否则,根据当前的赋值生成相应的短语(*generate_sat_induced_entity*()).该过程的基本思想为:当扩展合取范式中的每个子句的真值都为 1 时,可以从每个子句中任意选出一个文字,用合取符号把它们连接起来就生成了相应的短语.例如,合取范式$(x_1\vee\neg x_2)\wedge(\neg x_2\vee x_3)\wedge(x_3\vee x_4)$的一个满足赋值为$\{x_1=0, x_2=0,x_3=1,x_4=0\}$.可以从每个子句中挑选一个文字,假设选择了文字$\neg x_2$和 x_4,则生成的短语是$\neg x_2\wedge x_4$.显然可以看出,生成的短语不唯一.接下来,根据得到的短语反复进行短语学习过程(*consensus_generate_entity*()).最后,满足分析子过程把生成的短语加入到数据库中(*add_newentity*()),这实际上就是满足学习过程.判断新短语的决策层是否为 0:如果是,则算法结束;否则,回退到新短语所对应的决策层.这样,当随后的搜索过程中遇到同样的情况时,也可以不用继续搜索而直接判断当前的搜索分支是可满足的,进而直接搜索全称变量的另一个分支.它实现了算法的非顺序回退,使搜索向着一个有解空间进行,从而缩小了搜索空间.

```
Routine analyze_SAT()
1.  entity= find_sat_entity();
2.  if (entity= NULL)
3.  then entity = generate_sat_induced_entity();
4.  if (!termnate_condition(entity))
5.  then newen= consensus_generate_entity(entity);
6.  add_newentity(newen);
7.  if (dl(newen)≠0)
8.  then backtrack(dl(newen));
9.  end
```

Fig.7 Analyze SAT routine

图 7 满足分析子过程

5 实验比较

我们在 Linux 环境下用标准 C++语言实现了 QBF 问题求解器 HSPQBF 系统,其基本的算法即 HSPQBF 算法如图 2 所示.为了比较 QBF 求解器的求解效率,我们在有关 QBF 问题研究的网站上下载了目前公认的求解效率最高的几个 QBF 求解器,如 2ClsQ,SQBF,QuBErel1.3 等.所有的实验在一台 DELL PowerEdge 2650 上运行,实验的运行环境如下:CPU:2×Intel Xeon 2.00GHz,内存:1G,操作系统:RedHat ES3.0.

5.1 DPLL算法与HSPQBF算法比较

第1个实验主要考察第3节中介绍的Survey Propagation分支选择技术和第4节中介绍的冲突学习等分支处理技术在加速DPLL算法效率中的作用.在该实验中,所有的问题实例都是由著名的Gent & Walsh随机问题产生器生成的[13].

表1给出了具体的实验结果(时间单位是s,“—”表示在1 200s内执行失败的求解器),表中第1列中以1qbf,2qbf和3qbf开头的问题实例分别对应为1qbf问题、2qbf问题、3qbf问题,5cnf表示问题的每个子句的最大长度为5,160var表示最多有160个变量,2560cl表示子句最大个数为2 560个.我们用标准C++语言实现了图1中介绍的基于DPLL的QBF求解算法,表中的第2列给出了该算法的实验结果.第3列是在DPLL算法的基础上融入分支处理技术后的实验结果.可以看到,几乎在所有的问题上,算法的效率都有4倍以上的提高,对于一些问题(如3qbf-5cnf-20var-80cl.3),甚至有1 000倍以上的提高.第4列给出在DPLL的基础上融入调查传播技术后的实验结果,可以看到,这时在除了2qbf-5cnf-20var-80cl.0问题和3qbf-5cnf-20var-80cl.1问题外的几乎所有问题上,系统效率都有很大提高.第5列为最终系统的实验结果,可以看到,相对于DPLL算法,HSPQBF算法在所有的问题上效率都有所提高,对于很多问题,甚至有10^5规模以上的加速.

上述实验结果说明,基于Survey Propagation的分支选择技术和冲突学习等分支处理方法在提高DPLL算法的效率方面有着显著的效果.

Table 1 Comparisons of HSPQBF and DPLL in random problems

表1 HSPQBF与DPLL在随机问题上的实验比较

Problem	DPLL	DPLL+learning	DPLL+SP	HSPQBF
1qbf-5cnf-160var-2560cl.0	293.08	65.27	0.15	0.08
1qbf-5cnf-160var-2560cl.1	649.04	15.6	1.01	0.86
1qbf-5cnf-160var-2560cl.2	1014.52	25.56	0.08	0.04
1qbf-5cnf-160var-2560cl.3	—	176.41	0.09	0.13
1qbf-5cnf-160var-2560cl.4	11.15	0.45	1.76	0.94
1qbf-5cnf-160var-2560cl.5	—	—	0.7	0.37
1qbf-5cnf-160var-2560cl.6	904.43	6.22	0.12	0.06
1qbf-5cnf-160var-2560cl.7	250.63	12.7	0.04	0.02
1qbf-5cnf-160var-2560cl.8	374.41	68.84	11.87	12.06
1qbf-5cnf-160var-2560cl.9	7.19	0.73	3.82	5.9
2qbf-5cnf-20var-80cl.0	8.27	13.43	10.58	8.09
2qbf-5cnf-20var-80cl.1	2.75	1.38	2.17	1.72
2qbf-5cnf-20var-80cl.2	4.47	8.28	1.53	1.23
2qbf-5cnf-20var-80cl.3	3	1.68	0.65	0.51
2qbf-5cnf-20var-80cl.4	11.2	3.11	2.21	1.85
2qbf-5cnf-20var-80cl.5	4.92	0.86	2.41	1.89
3qbf-5cnf-20var-80cl.0	—	581.46	54.27	42.26
3qbf-5cnf-20var-80cl.1	227.52	8.52	265.65	163.08
3qbf-5cnf-20var-80cl.2	1007.78	636.82	847.08	629.66
3qbf-5cnf-20var-80cl.3	95.55	0.22	3.57	4.92
3qbf-5cnf-20var-80cl.4	—	908.35	349.51	354.24
3qbf-5cnf-20var-80cl.5	—	597.41	43.45	32.18

5.2 HSPQBF与参赛QBF求解器的效率比较

2ClsQ,SQBF,QuBErel1.3是目前公认的几个求解效率最高的QBF问题求解器,第2个实验是将HSPQBF系统和上述求解器进行比较.实验的测试用例包括由Gent & Walsh随机问题产生器生成的随机问题和我们在QBFLib[14]上下载的一些标准测试问题.

表2给出了上述4个系统在随机问题中的表现(时间单位是s,“—”表示在1 200s内执行失败的求解器).从表中可以清楚地看出,只有Quberel1.3和HSPQBF能够求解所有问题.除了在1qbf-5cnf-160var-2560cl.8问题中HSPQBF差于Quberel1.3外,在其他大部分问题中,HSPQBF的表现要远优于其他3个求解器.

表3给出了上述4个系统在Miscellanea域中的impl问题中的实验比对(单位是s,“—”表示在1 200s内执行失败的求解器).可以看出,除了2ClsQ外,其他3个系统都能够很快地解决上述问题.

表 4 给出了 4 个求解器对 Planning 域中的 ev-pr 问题求解的测试结果(单位是 s,“—”表示在 1 200s 内执行失败的求解器).从表中可以看出,HSPQBF 要明显好于 2ClsQ 和 Quberel1.3,但略差于 SQBF.

表 5 给出了 4 个 QBF 求解器对 Miscellanea 域中的 k_ph_p 问题求解的测试结果(单位是 s,“—”表示在 1 200s 内执行失败的求解器).从表中可以看出,HSPQBF 要明显好于其他 3 个求解器.在有些问题上,HSPQBF 甚至有 10 倍以上的加速.

Table 2 The solving effect list of random QBF problems

表 2 随机 QBF 问题的求解效果列表

Solver	2ClsQ	SQBF	Quberel1.3	HSPQBF
1qbf-5cnf-160var-2560cl.0	—	—	20.69	0.08
1qbf-5cnf-160var-2560cl.1	—	—	97.14	0.86
1qbf-5cnf-160var-2560cl.2	—	—	0.94	0.04
1qbf-5cnf-160var-2560cl.3	—	43.25	28.93	0.13
1qbf-5cnf-160var-2560cl.4	15.33	—	57.22	0.94
1qbf-5cnf-160var-2560cl.5	—	—	46.57	0.37
1qbf-5cnf-160var-2560cl.6	149.32	27	33.82	0.06
1qbf-5cnf-160var-2560cl.7	22.91	72.09	36.01	0.02
1qbf-5cnf-160var-2560cl.8	120.25	—	0.78	12.06
1qbf-5cnf-160var-2560cl.9	253.01	694.84	123.97	5.9
2qbf-5cnf-160var-5120cl.0	0.08	—	0.04	0
2qbf-5cnf-160var-5120cl.1	0.05	—	0.05	0.01
2qbf-5cnf-160var-5120cl.2	0.08	—	0.07	0
2qbf-5cnf-160var-5120cl.3	0.07	—	0.05	0.01
2qbf-5cnf-160var-5120cl.4	0.05	—	0.04	0
2qbf-5cnf-160var-5120cl.5	0.06	—	0.04	0
2qbf-5cnf-160var-5120cl.6	0.07	—	0.03	0
2qbf-5cnf-160var-5120cl.7	0.05	—	0.04	0.01
2qbf-5cnf-160var-5120cl.8	0.09	—	0.04	0
2qbf-5cnf-160var-5120cl.9	0.07	—	0.04	0

Table 3 The solving effect list of impl problems in the Miscellanea domain

表 3 Miscellanea 域中的 impl 问题的求解效果列表

Solver	2ClsQ	SQBF	Quberel1.3	HSPQBF
impl02	0	0	0	0
impl04	0	0	0	0
impl06	0.01	0	0	0
impl08	0.04	0	0	0
impl10	0.12	0	0	0
impl12	0.45	0	0	0
impl14	2.09	0	0	0
impl16	27.88	0	0	0

Table 4 The solving effect list of ev-pr problems in the Planning domain

表 4 Planning 域中的 ev-pr 问题的求解效果列表

Solver	2ClsQ	SQBF	Quberel1.3	HSPQBF
ev-pr-4x4-5-3-0-0-1-lg	—	1.55	1.06	1.22
ev-pr-4x4-7-3-0-0-1-lg	36.48	3.43	86.72	11.06
ev-pr-4x4-9-3-0-0-1-lg	547.45	5.79	—	117.08

Table 5 The solving effect list of k_ph_p problems in the Miscellanea domain

表 5 Miscellanea 域中的 k_ph_p 问题的求解效果列表

Solver	2ClsQ	SQBF	Quberel1.3	HSPQBF
k-ph-p-1	—	0	0	0
k-ph-p-2	0.01	0	0	0
k-ph-p-3	0.01	0.01	0	0
k-ph-p-4	0.02	0.07	0.02	0.01
k-ph-p-5	0.09	1.04	0.17	0.13
k-ph-p-6	0.73	4.15	1.29	0.45
k-ph-p-7	5.17	27.55	9.40	3.1
k-ph-p-8	76.52	653.04	206.57	16.78
k-ph-p-9	1 080.45	—	—	84.52

表 6 给出了 4 个 QBF 求解器对 Formal Verification 域中的 counter 问题求解的测试结果(单位是 s,“—”表示在 1 200s 内执行失败的求解器).从表中可以看出,只有 HSPQBF 和 SQBF 可以求解上述问题.其中,HSPQBF 要明显好于 2ClsQ 和 Quberel1.3,有些问题的求解速度甚至提高了近 300 倍,但略差于 SQBF.

Table 6 The solving effect list of counter problems in the Formal Verification domain

表 6 Formal Verification 域中的 counter 问题的求解效果列表

Solver	2ClsQ	SQBF	Quberel1.3	HSPQBF
cnt01	0	0	0	0
cnt02	0.01	0	0.01	0
cnt03	0.13	0.05	0.02	0.01
cnt04	11.81	0.07	0.49	0.04
cnt05	—	0.21	28.71	2.22
cnt06	—	1.27	—	0.98
cnt07	—	0.89	—	11.55
cnt08	—	3.46	—	79.96
cnt09	—	34.34	—	348.74
cnt10	—	43.64	—	237.86

综合上述两个实验可以看出,由于 Survey Propagation 方法确定 QBF 问题中的 backbone 变量主要依赖于计算变量赋值的概率边缘分布,所以得到正确的概率边缘分布显得尤为重要.对于 QBF 问题,随机问题实例的产生是由各个变量通过一定的概率形成的,因此,得到各个变量正确的概率边缘分布的几率更大;而结构化问题实例是通过实际问题转换产生的,因此,得到的各个变量的概率边缘分布往往有一定的错误,而错误的结果导致 Survey Propagation 方法不能提供有效的信息.所以,HSPQBF 算法在求解 QBF 随机问题实例上的表现更突出一些.另外,HSPQBF 不仅比原有的基于 DPLL 方法的 QBF 求解算法在效率上有了很大的提高,相对于目前最好的 QBF 求解器,同样具有竞争力.这是因为,对于大部分测试问题,SP 方法可以从局部信息的“陷阱”中跳出来,能够较早地找到 QBF 问题中的 backbone 变量,加速了 QBF 问题的简化,从而使得问题能够快速地得以解决.可见,SP 方法是可行的、有效的,因而在 DPLL 算法中融入 SP 方法后有效地提高了问题的求解效率.

冲突学习能够记录当前存在的冲突分支,当随后的搜索过程中遇到同样的情况时,可以不用继续搜索而直接判断当前的搜索分支是不可满足的.因而它能够对搜索空间进行剪枝,减小了搜索空间的大小.但是冲突学习的效率依赖于搜索过程中遇到的冲突数量,而在某些测试问题上,相应搜索树的叶节点几乎都是可满足的.在这种情况下,冲突学习在提高搜索效率上所起的作用很小.满足蕴含学习能够记录当前存在的满足分支,当随后的搜索过程中遇到同样的情况时,可以不用继续搜索而直接判断当前的搜索分支是可满足的,进而直接搜索全称变量的另一个分支.因而它同样缩小了搜索空间,进而提高了分支处理的效率.但该方法在有效减少搜索空间的过程中存在一定的时间消耗,这主要从两个方面体现出来:首先,在数据库中加入短语会降低单文字规则执行的速度;其次,产生的学习短语会花费一定的时间.因而在 DPLL 算法中融入分支处理技术后,算法的效率有了一定的提高,但加速的规模有限.因此从总体上讲,HSPQBF 算法在 DPLL 算法中融入 SP 方法以及分支处理等方法后有效地提高了 QBF 问题的求解效率.需要指出的是,在利用 Survey Propagation 方法确定 DPLL 分支时,步长对

启发式调查传播算法有着很大的影响,这主要体现在如下两个方面:

(1) 算法的效率;

(2) 算法的有效性.

Zecchina 等人指出,在求解 SAT 问题时,随着步长的递增,启发式调查传播算法的时间耗费快速递减,并且趋于平缓;同时,随着步长的递增,启发式调查传播算法的有效性快速递减.我们在求解 QBF 问题时发现,步长对系统的求解效率也有很大的影响,这主要体现在 SP 算法提供信息的量和信息的效率两方面上.在上述实验中,经过效率比对,将步长统一设置为 30.事实上,如何确定步长的大小,对于我们依然是一个悬而未决的问题.在此,将其留为一个开放式问题.

6 总 结

本文中,我们提出了一种启发式调查传播算法,并基于此算法设计了一种新的 QBF 求解器——HSPQBF 系统.这种求解器在分支选择时应用了调查传播方法来确定 DPLL 分支的顺序;在处理分支时,该系统利用冲突推理、冲突学习和满足蕴涵学习等方法对搜索空间剪枝,从而达到提高系统效率的目的.实验结果表明,调查传播方法作为一种启发式方法,可以有效地确定具有偏向性的变量,进而简化公式.冲突学习等技术也可以有效地加速 QBF 问题的求解.在大部分标准测试问题上,HSPQBF 要优于现有的 QBF 问题求解系统.然而,虽然本文提出的启发式调查传播算法可以减小搜索空间,并减少算法回退的次数,但为了证明 Survey Propagation 方法在 QBF 问题中的有效性,我们还需要有更进一步的研究.比如从统计物理学中的自旋玻璃原理入手,研究自旋玻璃与 QBF 问题的关系,在理论上说明该方法在 QBF 问题中的有效性.

References:

[1] Garey MR, Johnson DS. Computers and Intractability, A Guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman, 1979.

[2] Pan GQ, Vardi MY. Optimizing a BDD-based modal solver. In: Baader F, ed. Proc. of the 19th Int'l Conf. on Automated Deduction. Berlin, Heidelberg: Springer-Verlag, 2003. 75−89. [doi: 10.1007/978-3-540-45085-6_7]

[3] Rintanen J. Asymptotically optimal encodings of conformant planning in QBF. In: Collins J, Faratin P, Parsons S, eds. Proc. of the 22nd AAAI Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 2007. 1045−1050.

[4] Egly U, Eiter T, Tompits H, Woltran S. Solving advanced reasoning tasks using quantified Boolean formulas. In: Collins J, Faratin P, Parsons S, eds. Proc. of the 12th AAAI Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 2000. 417−422.

[5] Gent IP, Hoos HH, Rowley AGD, Smyth K. Using stochastic local search to solve quantified Boolean formulae. In: Rossi F, ed. Proc. of the Principles and Practice of Constraint Programming. Berlin, Heidelberg: Springer-Verlag, 2003. 348−362.

[6] Gent I, Giunchiglia E, Narizzano M, Rowley A, Tachella A. Watched data structures for QBF solvers. In: Giunchiglia E, Tacchella A, eds. Proc. of the 6th Int'l Conf. on Theory and Applications of Satisfiability Testing. LNCS 2919, Berlin, Heidelberg: Springer-Verlag, 2003. 348−355. [doi: 10.1007/978-3-540-24605-3_3]

[7] Giunchiglia E, Narizzano4 M, Tacchella A. QuBE: A system for deciding quantified Boolean formulas satisfiability. In: Nebel B, ed. Proc. of the 17th Int'l Joint Conf. on Artificial Intelligence. LNCS 2083, Morgan Kaufmann Publishers, 2001. 18−23. [doi: 10.1007/3-540-45744-5_27]

[8] Rintanen J. Partial implicit unfolding in the Davis Putnam procedure for quantified Boolean formulae. In: Nieuwenhuis R, Voronkov A, eds. Proc. of the 8th Int'l Conf. on Logic for Programming, Artificial Intelligence and Reasoning. LNCS 2250, Berlin, Heidelberg: Springer-Verlag, 2001. 362−376. [doi: 10.1007/3-540-45653-8_25]

[9] Zhang LT, Malik S. Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation. In: Rossi F, ed. Proc. of the Principles and Practice of Constraint Programming. Berlin, Heidelberg: Springer-Verlag, 2002. 200−215. [doi: 10.1007/3-540-46135-3_14]

[10] Cadoli M, Giovanardi A, Schaerf M. An algorithm to evaluate quantified Boolean formulae. In: Collins J, Faratin P, Parsons S, eds. Proc. of the AAAI Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 1998. 262−267.

[11] Monasson R, Zecchina R, Kirkpatrick S, Selman B, Troyansky L. Determining computational complexity from characteristic ‘phase transitions’. Nature, 1999,400(6740):133−137. [doi: 10.1038/22055]

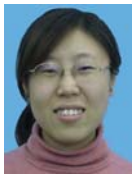
[12] Mézard M, Parisi G, Zecchina R. Analytic and algorithmic solution of random satisfiability problems. Science, 2002, 297(5582):812−815. [doi: 10.1126/science.1073287]

[13] Gent I, Walsh T. Beyond NP: The QSAT phase transition. In: Collins J, Faratin P, Parsons S, eds. Proc. of the AAAI Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 1999. 648−653.

[14] Giunchiglia E, Narizzano M, Tacchella A. Quantified Boolean formulas satisfiability library (QBFLIB). 2001. http://www.qbflib.org/

殷明浩(1979－),男,安徽安庆人,博士,副教授,CCF 会员,主要研究领域为智能规划,自动推理.

孙吉贵(1962－2008),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能规划,自动推理.



周俊萍(1981－),女,博士,主要研究领域为智能规划.

谷文祥(1947－),男,教授,博士生导师,主要研究领域为智能规划与规划识别,模糊数学及其应用.