H-GRASP：一种基于 GRASP 改进的混合 SAT 解法器*

唐玉兰[1]，张惠国[1]，于宗光[1,2]，陈建慧[1,3]

（1. 江南大学 信息工程学院，江苏 无锡 214122；2. 中国电子科技集团公司 第五十八研究所，江苏 无锡 214035；3. 无锡职业技术学院，江苏 无锡 214121）

摘 要：为了改善 GRASP 的局限性，提出了一种能解决含有伪布尔（PB）和合取范式（CNF）混合约束问题的新的混合算法（H-GRASP）。该新算法采用了切削平面技术来提取 PB 约束条件之间的推论，并把它结合到普通的蕴涵图中，分析引起冲突的学习。与解决混合约束问题的其他两种方法——整数线性规划和纯基于 SAT 方法进行了彻底的比较。实验结果证明，H-GRASP 方法从整体上大大减少了运行时间，加快了速度，同时还保证了加入这种方法的低耗费。

关键词：布尔可满足性；势约束；整数线性规划；伪布尔

中图分类号：TN405.97 **文献标志码**：A **文章编号**：1001-3695（2010）03-0864-04

doi：10.3969/j.issn.1001-3695.2010.03.015

H-GRASP：advanced GRASP-based hybrid SAT solver

TANG Yu-lan[1], ZHANG Hui-guo[1], YU Zong-guang[1,2], CHEN Jian-hui[1,3]

(1. *School of Information Technology, Southern Yangtze University, Wuxi Jiangsu* 214122, *China*; 2. *No.* 58 *Research Institute, China Electronics Technology Group Corporation, Wuxi Jiangsu* 214035, *China*; 3. *Wuxi Institute of Technology, Wuxi Jiangsu* 214121, *China*)

Abstract: In order to improve the limitations of GRASP, this paper proposed a new hybrid algorithm (H-GRASP), which could resolve the problem of mixed constraints of pseudo-Boolean (PB) and conjunctive normal from (CNF). The new method adopted the cutting-plane technique to draw inferences among PB constraints and combined it with generic implication graph analysis for conflict-induced learning. Presented a thorough comparison of the new technique against the other two methods-integer linear programming (ILP) and pure SAT-based methods also. Experimental results prove that H-GRASP has significantly reduced running time and sped up, while keeping the overhead of adding it into the problem low.

Key words: Boolean satisfiability(SAT); cardinality constraint; integer linear programming; pseudo-Boolean(PB)

0 引言

近年来，随着解决布尔可满足性（SAT）方法的重大发展和对伪布尔（PB）约束的几种扩展，SAT 出现在集成电路电脑辅助设计的许多领域，包括测试向量自动生成、时序分析、延时故障测试、逻辑验证和 FPGA 布局布线等。SAT 属于经典的 NP-完全性问题，在最坏情况下，其复杂度随指数级增长。多年来，已经为 SAT 提出了很多成功的求解程序，如 GRASP[1]、GRAPE[2]、Chaff[3]，SAT[4]。这些程序大多属于 DPLL（Davis Putnam Logemann Loveland）算法[5]的变异型，即为一类查找型算法。

GRASP 是一种针对提出的可满足性问题的一种综合的基于查找的 SAT 算法构架。这一框架的前提是在搜索过程中冲突的不可避免性。GRASP 结合了几种搜索修剪技术，被证明在解决各种 SAT 问题时是相当强大的。其最显著特点是用强大的冲突分析过程增加了基本回溯搜索。冲突分析确定使之非顺序回溯到搜索树中先前水平，潜在地减少了大部分的搜索空间。然而合取范式（conjunctive normal from，CNF）约束虽然在大范围的应用中表现很好，但并不总是最有效的。从逻辑综合验证[6]到很多操作研究应用，这些问题都可以用 PB 约束来更有效地进行编译。另外，PB 约束在表达目标函数的优化应用方面也有很好的运用，如 Max-SAT 和 Max-ONEs[7]。为了在搜索过程中减少空间，很多解法器都动态地引入了新的约束。在整数线性规划（integer linear programming，ILP）中，通过在分支和边界程序中进行切削平面分析[8]来产生这样的约束。换言之，现在的 SAT 和 PB 解法器中的约束是通过分析一条 CNF 子句中每个导致冲突的蕴涵图[9]来产生的，或者是通过切削平面分析产生一条 PB 约束或一条势约束[10]（cardinality constraint）来产生的。在文献[11]中切削平面还可用于 PB 优化程序中，用来计算目标函数的更低的边界。

本文针对 GRASP 的这种局限性，提出了一种解决含有 PB 和 CNF 混合约束问题的新算法，这种新算法是结合了切削平

收稿日期：2009-07-24；**修回日期**：2009-09-09 **基金项目**：江苏省自然科学基金资助项目（BK2007026）；江苏省“333 高层次人才培养工程”专项资助项目（2007124）

作者简介：唐玉兰（1981-），女，江苏丹阳人，博士研究生，主要研究方向为 FPGA 的布局布线（shaoyi323@126.com）；张惠国（1978-），男，江苏常熟人，博士研究生，主要研究方向为数字集成电路设计；于宗光（1964-），男，山东维坊人，教授，博导，主要研究方向为大规模集成电路设计；陈建慧（1980-），男，江苏丹阳人，硕士研究生，主要研究方向为计算机应用.

面方法和基于冲突学习的进行蕴涵图分析的方法。这样,学习程序就结合了蕴涵图有效识别单位子句方面的优势,也结合了通过切削平面程序学习 PB 约束方面的优势。因此,在发现一个冲突后,采用了同时构造一条 PB 约束和一条 CNF 子句的方法。只要 PB 约束还是活动的,这种策略使解法器可以完全利用 PB 约束减少空间能力。当 PB 约束变为静止时,就会被丢弃,它的任务就由其同伴 CNF 子句来承担,这样就更进一步控制了其耗费。

1　预备知识

1.1　PB 约束的定义

一个普通形式的 PB 约束可以表示为

$$\sum_{i=1}^{n} a_i \dot{x}_i \geq b \tag{1}$$

其中:$a_i, b \in Z^+$;$\dot{x}_i$ 代表 x_i 或 $\bar{x}_i$。如果对它的相关变量进行当前赋值,$\dot{x}_i$ 等于 1,那么 $\dot{x}_i$ 就是一个真文字。约束中最大的系数表示为 $a_{\max}$,它所对应的文字表示为 $\dot{x}_{\max}$。

下面是对 PB 约束所作的一些简化处理,在求解之前和当中都需要用到它们。

系数简化[12]:如果 $a_k > b, k \in \{1, \cdots, n\}$,式(1)中所示的正常形式的 PB 约束就相当于下面的 PB 约束。

$$b\dot{x}_k + \sum_{i=1, j \neq k}^{n} a_i \dot{x}_i \geq b \tag{2}$$

弱化势约束:PB 约束式(1)的最强大的扩展子句是较弱的势约束。

$$\sum_{i=1}^{n} \dot{x}_i \geq \beta \tag{3}$$

这里 β 的是最小的整数,如

$$\sum_{i=1}^{\beta-1} a_i < b \leq \sum_{i=1}^{\beta} a_i \tag{4}$$

1.2　PB 约束求解

1.2.1　整数线性规划法

这是应用于 CPLEX[13] 等几个商业工具的方法,是基于松弛 0-1 变量的完整性,并用单一程序来解决分支限界(branch-and-bound, BNB)树中每个节点所带来的 LP 问题,一旦找到一个整数解或者所有的节点都被检查过后,搜索过程结束。这种方法的最近发展是由于使用了分支切割(branch-and-cut)技术,这项技术可以找到在 0-1 问题中有效的不等式,但是违反了 LP 松弛。因此,把这些不等式加到 LP 松弛中去固定了公式,加强了 BNB 结构。

1.2.2　纯基于 SAT 的方法

解决一个混合了 CNF 子句的 PB 约束问题的最简单方法就是把它们转换成相等的 CNF 子句问题并传送给 SAT 解法器,如 MiniSAT + [14]。一种把 PB 转换成 CNF 的方法就是,把每条 PB 约束用电路代表,产生一条 CNF 公式[9]。下面是把 PB 约束转换成 CNF 约束的三种技术:

a)把 PB 约束转换成一个 BDD,再把代替 PB 约束的 BDD 转换成 CNF 约束。

b)把 PB 约束转换成一个加法器网络,PB 约束左手边的和作为一个二进制编码,构造一个电路代表和与其右手边的比较。这个电路的 CNF 表述就是 PB 约束的转换。

c)把 PB 约束转换成一个分类网络,然后构造比较电路并转换成 CNF 子句。

要了解这些方法的更多细节,可以参考文献[14]。

2　混合方法(H-GRASP)

用扩展的 DLL-类型的 SAT 解法器来处理 PB 约束[15],是 PB 约束求解最普遍的基于 SAT 的方法,这种方法是把 PB 约束和 CNF 子句作为单独的约束类别来看待。把 PB 约束结合到 SAT 解法器的几种技术和方法在最近的几年里又有了发展。文献[14]采用了看守所有文字策略和 CNF 学习方法来进行 PB 传输和学习。文献[16]第一次把基于切削平面引起冲突的学习引入到 SAT 解法器中。文献[10]认识到处理很多产生的 PB 约束(切削平面)的效率很低,所以还引入了一种弱化方法,用于所有产生的切削平面(后简化),把它们转换成势约束。文献[17]还提出了一种周期性地丢弃那些约束的方法,从而可以解决产生的很多切削平面的负担。

本章提出了一种新的混合方法(H-GRASP),基于 GRASP 作了一些改进,用来分析蕴涵图并产生切削平面,在每个冲突时同时产生一条 CNF 子句和一条 PB 约束,并讨论了有效处理 PB 约束的方法。

2.1　引入混合学习

认识到 CNF 和 PB 学习各自的优点,CNF 可以很方便地运用并总能产生一个单位约束,而 PB 学习可以减少能源消耗。提出的混合算法有效地结合了 CNF 和 PB 学习方法,在冲突时同时产生一条 CNF 子句和一条 PB 约束。在这种混合方法中,CNF 学习通过产生一条单位 CNF 子句来保证程序的正确性和完整性,一旦到达第一个 UIP,学习过程终止。另外,同时产生一条 PB 约束可以大大减少搜索空间。这可以与文献[10]中的方法相比较,学习过的 PB 约束被弱化成势约束,然后用来结束学习过程,并保留作为最后的学习过的约束。可以采用文献[17]中提出的方法在冲突时产生并增加新的 PB 约束,其过程如下:

```
bool PB_new(Solver S, int goal, Vec<lit_coef> pbs, PseudoBoolout,
Clause c, bool learnt)
    out = newPseudoBool
    out.rhs = goal
    for(int i=0; i<pbs.size(); i++)
      out.terms[i].lit = Lit(pbs[i], lit)
      if(pbs[i].coef < o)
        out.rhs += abs(pbs[i].coef)
        out.terms[i].lit = out.terms[i].lit
      out.terms[i].coeff = abs(pbs[i].coef)
    sart(out.terms, size())
    if((out.terms[0].coeff == out.rhs &&out.terms[size()].
coeff == out.rhs) ||
      out.rhs == 1) - checking if PB constraint is equal to a CNF clause
      bool ret = conver PB to CNF(s, out, c, learnt) - creates clause with
literals in terms of out
      xfee(out)
      return ret
    out. amax = out.terms[0].coef
    out.learnt = learnt
    if(out.rhs == 0) return true
    for (int i=0; i<size(); i++) - setting up the watch list
      Lit lit = out.terms[i].lit
      if (out.watchsum < out.rhs + out.amax) out.terms[i].watch(S,
```

```
this)
    if ( out. watchsumout. rhs + out. amax) break
  if ( out. watchsum < out. rhs) return false
  if ( out. watchsum < out. rhs + out. amax)
    for ( int i = 0; i < size( ); i ++ )
      Lit lit = out. terms[ i ]. lit
      if ( S. value( lit) = = 1_Undef)
        if ( out. watchsum out. terms[ i ]. coeff + out. rhs) break
        if ( ! S. enqueue( lit)) return false
    if ( learnt) S. var Bump Activity( lit, coeff/out. rhs)
    return true
```

认识到每个解决步骤中预解式 R_i 中的假文字代表了蕴涵图在这个步骤的切割,这是很重要的。因此它们储存于一个独立的组 X_F,用做 CNF 学习。这就使 CNF 学习有高效率,并显著最小化其耗费。

当到达第一个 UIP 时,整个学习过程终止。UIP 是在 CNF 学习过程中自动搜索的。从这一点来说,储存在 X_F 中的学习过的 CNF 子句是单位的,而预解式 R_i 组成了学习过的 PB 约束。回溯等级是通过处理相应条件下的学习过的 CNF 和 PB 约束得到的。

在这个混合算法中,没有必要消除学习过的 PB 约束的过份满足,或者是通过一个弱化步骤来使其变成单位形式。因为根据经验发现,在当前的真分配中,对它们进行弱化有利于在一定程度上增加其修剪能力。为了减少这种弱化的耗费,可以利用蕴涵图的结构来代替过份满足的单位 PB 约束,即用 CNF 子句代表约束所表示的对应边的切割。

2.2　布尔约束传播(Boolean constraint propagation,BCP)

PB 约束的 BCP 是基于以下思想:一旦一个文字的系数必须包含在满足的约束中,那么这个文字是隐含的。对形式分析而言,用 s 表示约束的松弛,如下所示:

$$s = \sum_{l_i \neq \text{false}} a_i - k \tag{5}$$

这里的 k 表示约束的度。非正式的说法就是,假如所有当前的未赋值的文字最终都为真,松弛就是约束可以被过份满足的最大数。于是,对于任何未赋值的文字 l_i,都有

$$s - a_i < 0 \tag{6}$$

l_i 在当前变量分配下是隐含的。

学习 PB 约束的最主要缺点就是传输约束的高传输耗费[10],这可能会由于它减少了搜索空间而抵消。本节为 PB 约束提出了看守策略,来看守 PB 约束中足够多的文字,使 $S_W \geqslant b + a_{\max}$,这可以使传输变得很快。此项技术是由文献[10]中所提出的,但是并没有运用在无限制的 PB 约束中。

下面描述了怎样把扩展看守文字策略用于 PB 约束。看守文字的思想是最小化约束的监控工作,当一条约束被违反或者其中的一些未赋值文字中被隐含赋值时,仍然能侦测出精确的时间。为了保证不违反约束,可以看守一组真或者未赋值的文字 L_w,如下所示:

$$\sum_{i \in L_w} a_i = S_w \geqslant k \tag{7}$$

其中:S_w 表示看守总数。这是因为所有的 L_w 文字仍然是尚待证实的,满足约束。

侦测隐含的赋值需要进行预行估计。为了保证没有隐含变量赋值,就需要看守一组真或者未赋值的文字 L_w,如下所示:

$$\sum_{i \in L_w} a_i = S_w \geqslant k + a_{\max} \tag{8}$$

其中:$a_{\max}$ 表示任意未赋值文字(或者一些更大数)的最大系数,这就保证了不存在未赋值的文字 l_i,此时

$$a_i > s \geqslant (S_w - k) \geqslant a_{\max}$$

PB 约束的 BCP 算法如下:

Algorithm 1 BCP triggered by watched literal l_t = false
L_w: Set of watched literals of constraint
L_u: Set of non-watched, non-false literals of constraint
$\{a_i\}$: Set of literal coefficients in L_w and L_u
S_Q: Watch Sum
a)　$L_w \leftarrow L_w \setminus \{l_t\}$
b)　$S_w \leftarrow S_w - a_t$
c)　$a_{\max} \leftarrow \max\{a_i \mid l_i \in L_w \cup L_u \wedge l_i \neq \text{true}\}$
d)　while $S_w < k + a_{\max} \wedge L_u \neq 0$ // fill watch set
e)　　$a_s \leftarrow \max\{a_i \mid l_i \in L_u\}$
f)　　$S_w \leftarrow S_w + a_s$
g)　　$L_w \leftarrow L_w \cup \{l_s\}$
h)　　$L_u \leftarrow L_u \setminus \{l_s\}$
i)　if ($S_w < k$) // detect conflict
j)　　return CONFLICT
k)　while $S_w < k + a_{\max}$// detect implications
l)　　IMPLY ($l_{\max}$)
m)　　$a_{\max} \leftarrow \max\{a_i \mid l_i \in L_w \wedge l_i \neq \text{true}\}$
n)　return NO_CONFLICT

只要 L_w 的一个文字赋值为假,它就将从 L_w 中移除,加入一个新的真或者未赋值的文字到 L_w 中,直到满足条件 h)。如果这不能完成,就继续隐含 L_w 中的文字满足 f),减少 $a_{\max}$,直到最终满足 h)。上述算法是 PB 约束的 BCP 处理过程,这是由看守文字赋值为假引起的。

3　实验结果比较及分析

所有的实验都是在 1 GB RAM、2 GHz Intel 双核下运行的,操作系统为 Linux RedHat,时间限制为 1 000 s。分别从全局布线(grout)[18]、FPGA 布线(fpga)[19]和 DIMACS[20]系列中各自选择了一些的例子。文献[21]中报告了在这些基准下各种解法器运行的详细结果。对 H-GRASP 混合算法和其他求解方法进行比较,即 ILP 方法和转换成 SAT 这两种方法。下面采用不同的学习和传输策略比较混合算法和其他的处理方法。为此,用 PB'05 评估的结论[21]给出了一个全面的分析。在一组代表性的基准上,用三种不同的方法来解决 PB 约束,即 ILP(用 SATIRE[22])、SAT 的预处理(用 MiniSAT + [14])和 PB-SAT(用 H-GRASP),其结果如表 1 和 2 所示,最佳结果用粗体表示。

表 1　全局布线实例的尺寸和运行时间

实例	实例尺寸				时间/s		
名称	线网	变量	CNF	PB	SATIRE	MiniSAT +	H-GRASP
grount-3.3-1	18	216	572	12	**0.41**	3.25	1.56
grount-3.3-2	22	264	700	12	0.96	1.15	**0.32**
grount-3.3-3	20	240	636	12	1.1	1.63	**0.09**
grount-3.3-4	19	228	604	12	**0.2**	1.30	1.16
grount-3.3-5	20	240	634	12	**0.35**	1.50	0.82
grount-4.3-1	28	672	2 004	24	109.7	254	**3.66**
grount-4.3-2	27	648	1 928	24	32.13	203	**1.72**
grount-4.3-3	27	648	1 930	24	319.47	145	**5.5**
grount-4.3-4	29	696	2 072	24	>1 000	74.4	**16.4**
grount-4.3-5	30	720	2 144	24	567.12	>1 000	**27.2**
grount-4.3-6	26	624	1 860	24	>1 000	117	**28**
grount-4.3-7	28	672	2 006	24	>1 000	>1000	**65**
grount-4.3-8	18	432	1 280	24	177.8	60	**3.9**
grount-4.3-9	35	840	2 502	24	>1 000	271	**56**
grount-4.3-10	38	840	2 504	24	>1 000	159	**7.5**

由表 1 中可以得出,混合方法比纯基于 SAT 方法有效,它

利用了问题的切削平面和 PB 结构。基于切削平面理论的学习方法可以减少在解决问题时 ILP 与基于 SAT 的方法之间的运行差距,所以可以采用 ILP 框架更有效地解决那些问题,而无须使用基于 SAT 技术,它会带来过份约束的问题。特别是在表 2 的 FPGA 的布线基准中,混合处理方法表现出比其他两种方法相当大的优势。注意这些基准对于这样的混合解法器特别好,因为其输入也是混合的。

表 2　FPGA 布线和 DIMACS 实例的尺寸和运行时间

基准组	可满足的实例			时间/s		
	名称	变量	子句	SATIRE	MiniSAT +	H-GRASP
FPGA 布线	fpga35-33sat	1 733	1 256	10.39	>1 000	0.2
	fpga35-35sat	1 838	1 330	12.6	53.452	0.25
	fpga40-39sat	2 340	1 678	36.89	203.43	0.78
	fpga40-40sat	2 400	1 720	49.53	468.60	0.33
	fpga45-44sat	2 970	2 113	314.4	>1 000	2.77
DIMACS	aim-50-16-y1-1	50	80	0.011	0.012	0.01
	aim-100-16-y1-1	100	160	0.02	0.013	0.01
	aim-200-20-y1-1	200	400	0.06	0.026	0.12
	ii8b1	336	2068	>1 000	23.25	682.03
	jnh1	100	850	2.2	2.15	1.97
	par8-1	350	1149	0.06	0.042	0.04

值得一提的是,基于 SAT 的方法可以通过运用文献[23]中更低的边界计算的方法,使其在优化问题上的性能得到进一步的提高。而这样的局部估算在 ILP 解法器中是常见现象,有利于巩固其性能。

4　结束语

本文针对 GRASP 不能应用于混合约束的局限,引入了基于切削平面的解决混合约束的学习方法,提出了一种新的混合学习方法(H-GRASP)。继承了 GRASP 的优点以外,还采用了一个混合程序,可以在每个冲突之后产生一条 PB 和一条 CNF 约束。这样的双重学习方法很大程度地提高了求解速度,达到了其他算法无法达到的好成绩。在进一步的工作当中主要研究如何进一步控制 PB 约束高耗费的问题,另外,更为高效简洁的算法也是研究的主要目标。

参考文献:

[1] SILVA J, SAKALLAH K. GRASP: a search algorithm for propositional satisfiability[J]. IEEE Trans on Computers, 1999,48(5): 506-521.

[2] SOICHER L H. GRAPE: a system for computing with graphs and groups[J]. DIMACS Ser in Discr Math & Theor Comp Sci, 1993,11:287-291.

[3] MOSKEWICI M W, MADIGAN C F,ZHAO Ying, *et al*. Chaff: engineering an efficient SAT solver[C]//Proc of the 39th Design Automation Conference. 2001:530-535.

[4] NAM G J, SAKALLAH K, RUTENBAR R A. A Boolean satisfiability based incremental rerouting approach with application to FPGAs [C]//Proc of Design Automation & Test Europe. 2001: 560-565.

[5] BEAME P, KARP, PITASSI T. The efficiency of resolution and Davis-Putnam procedure[J]. SIAM Journal on Computing, 2002,4 (31):1048-1057.

[6] BRYANT R E, LAHIRI S K, SESHIA S A. Deciding CLU logic formulas via Boolean and pseudo-Boolean encodings[C]//Proc of Constraints in Formal Verification. 2002.

[7] CREIGNOU N, KHANNA S, SUDAN M. Complexity classifications of Boolean constraint satisfaction problems[C]//Proc of Siam Monographs on Discrete Mathematics and Applications. 2001:106.

[8] GOMORY R E. An algorithm for integer solutions to linear programs [C]//Proc of Recent Advances in Mathematical Programming. 1963: 269-302.

[9] ALOUL F A, RAMANI I, MARKOV L, *et al*. Generic ILP versus specialized 0-1 ILP[C]//Proc of IEEE/ACM International Conference on Computer-Aided Design. 2002: 450-457.

[10] KUEHLMANN C D A. A fast pseudo-Boolean constraint solver[J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2005,24(3):305-317.

[11] MANQUINHO V M, MARQUES-SILVA J P. On applying cutting planes in DLL-based algorithms for pseudo-Boolean optimization [C]//Proc of the 8th International Conference on Theory and Applications of Satisfiability Testing . 2005:451-458.

[12] CROWDER H, JOHNSON E, PADBERG M W. Solving large-scale zero-one linear programming problems [J]. Operations Research, 1983,31(5):803-834.

[13] ILOG Inc. CPLEX version 9[EB/OL]. http://ilog.com/products/cplex/.

[14] EÉN N, SÖRENSSON N. Translating pseudo-Boolean constraints into SAT[J]. Journal on Satisfiability, Boolean Modeling and Computation, 2006,2:1-26.

[15] BARTH P. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization, Technical Report MPI-I-95-2-003[R]. 1995.

[16] DIXON H E, GINSBERG M L. Inference methods for a pseudo-Boolean satisfiability solver[C]//Proc of the 18th National Conference on Aritificial Intelligence. 2002: 635-640.

[17] SHEINI H M,SAKALLAH K A. Pueblo: a hybrid pseudo-Boolean SAT solver[J]. Boolean Modeling and Computation, 2006,2: 165-189.

[18] ALOUL F A, MARKOV I L, SAKALLAH K A. Solving difficult instances of Boolean satisfiability in the presence of symmetries[J]. IEEE Trans on Computer Aided Design, 2003,22(9): 1117-1137.

[19] NAM G J, SAKALLAH K, RUTENBAR R A. A Boolean satisfiability-based incremental rerouting approach with application to FPGAs [C]//Proc of Design Automation & Test Europe. 2001:560-565.

[20] DIMACS challenge benchmarks[EB/OL]. ftp://Dimacs.rutgers.EDU/pub/challenge/sat/benchmarks/cnf.

[21] MANQUINHO V M, ROUSSEL O. The first evaluation of pseudo-Boolean solvers (PB05)[J]. Journal on Satisfiability, Boolean Modeling and Computation, 2006,2:103-143.

[22] WHITTENMORE J, KIM J, SAKALLAH K. SATIRE: a new incremental satisfiability engine[C]//Proc of Design Automation Conference. 2001: 542-545.

[23] MANQUINHO V M, MARQUES-SILVA J P. Effective lower bounding techniques for pseudo-Boolean optimization[C]//Proc of Design, Automation and Test in Europe. 2005:660-665.