# 伪布尔约束的一种模型计数方法

郑苏豪, 牛秦洲, 陶小梅

引用本文

郑苏豪, 牛秦洲, 陶小梅. 伪布尔约束的一种模型计数方法[J]. 计算机科学, 2024, 51(11A): 240300161-5.

ZHENG Suhao, NIU Qinzhou, TAO Xiaomei. Model Counting Method for Pseudo-Boolean Constraints [J]. Computer Science, 2024, 51(11A): 240300161-5.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240300161

# 伪布尔约束的一种模型计数方法

郑苏豪[1]　牛秦洲[1]　陶小梅[2]
1 桂林理工大学信息科学与工程学院　广西 桂林 541006
2 广西师范大学计算机科学与工程学院、软件学院　广西 桂林 541006
(1659792900@qq.com)

**摘　要**　伪布尔约束问题是一类与布尔约束问题相似的组合优化难题。解决这类问题的核心在于以不同的数学形式对伪布尔约束进行编码,例如线性规划、整数规划以及其他形式的组合优化。当前流行的解决方法是将问题转化为布尔公式,然后运用冲突导向的子句学习(CDCL)类 SAT 求解器对这些布尔公式进行求解。提出了一种新的方法来解决伪布尔约束问题中的模型计数问题。首先,介绍了知识编译和扩展规则的相关概念,随后详细阐述了如何利用知识编译将伪布尔约束问题转化为二元决策图(BDD),并着重探讨了 BDD 结构的特性,最后采用基于扩展规则的模型计数方法来处理伪布尔约束问题中的模型计数问题。实验结果表明,该方法在处理互补因子较高的子句集时表现出更为优越的性能。

**关键词**:伪布尔约束;SAT 问题;模型计数;知识编译;子句
**中图分类号**　TP301

## Model Counting Method for Pseudo-Boolean Constraints

ZHENG Suhao[1], NIU Qinzhou[1] and TAO Xiaomei[2]
1 College of Information Science and Engineering, Guilin University of Technology, Guilin, Guangxi 541006, China
2 School of Computer Science and Engineering & School of Software, Guangxi Normal University, Guilin, Guangxi 541006, China

**Abstract**　Pseudo-Boolean constraints problem is a kind of combinatorial optimization problem similar to the Boolean constraints problem. The core of solving such problems lies in encoding Pseudo-Boolean constraints in different mathematical forms, such as linear programming, integer programming, and other forms of combinatorial optimization. The current popular solution is to convert the problem into Boolean formulas, and then use the conflict-oriented clause learning(CDCL) class SAT solver to solve these Boolean formulas. This paper proposes a new method to solve the model counting problem in the Pseudo-Boolean constraints problem. Firstly, the related concepts of knowledge compilation and extension rules are introduced, and then how to transform Pseudo-Boolean constraints problem into binary decision diagram(BDD) by knowledge compilation is discussed in detail, and the characteristics of BDD structure are discussed emphatically. Finally, the model counting method based on extension rules is adopted to deal with the model counting problem in Pseudo-Boolean constraints problem. Experimental results show that the proposed method has better performance when dealing with clauses with higher complementarity factors.

**Keywords**　Pseudo-Boolean constraints, SAT problem, Model counting, Knowledge compilation, Clause

## 1　引言

离散决策问题是生活中比较常见的一类问题,这类问题通常可以表示为约束满足问题(Constraint Satisfaction Problems,CSPs)或约束优化问题(Constrained Optimization Problems,COPs)。线性方程和不等式在诸如调度、路由、资源分配和许多其他难组合问题等约束问题中无处不在。伪布尔约束(Pseudo-Boolean Constraints,PB)即形式为 $a_1x_1+\cdots+a_nx_n \# K$ 的特殊约束,其中 $a_i$ 和 $K$ 为整数系数,$x_i$ 为布尔变量,关系运算符 $\# \in \{<,>,\leqslant,\geqslant,=\}$。通常假设 $\#$ 取 $\leqslant$,$a_i$ 和 $K$ 取正数,其他情况可以在多项式时间内转换为这种形式[1]。目前求解 PB 约束的方法大多是将它们编码为布尔公式转化为 SAT 问题,然后应用 CDCL(Conflict-Directed Clause Learning)求解器求解[2-3]。因而 PB 问题和 SAT 问题的研究也越来越密切,如 Ding 等提出的将 SAT 问题转化为 PB 约束并使用单纯形法进行求解的方法[4]以及 Fang 等提出的将 1-in-3 SAT 问题转化为线性约束系统进行求解以实现对 SAT 求解的方法[5]等。本文设计了一种伪布尔约束模型计数方法,利用知识编译将伪布尔约束转化为二元决策图(Binary Decision Diagram,BDD),将伪布尔约束模型计数转化 # SAT 问题,再利用相关算法进行求解。

## 2　相关定义

**定义 1**(扩展规则)[6]　给定子句 $C$ 和原子的集合 $M$:

基金项目:国家自然科学基金(61906051)
This work was supported by the National Natural Science Foundation of China(61906051).
通信作者:牛秦洲(niuqinzhou@163.com)

$D=\{C\vee a, C\vee\neg a \mid a\in M\}$，把从 $C$ 到 $D$ 中元素的推导过程叫做扩展规则，$D$ 中的元素叫做应用扩展规则的结果。

例 1 给定子句 $x_1\vee x_2$ 和变量集 $\{x_1, x_2, x_3\}$，对子句 $x_1\vee x_2$ 应用扩展规则得到 $x_1\vee x_2\vee x_3$ 和 $x_1\vee x_2\vee\neg x_3$。

**定义 2**(极大项)[7] 一个子句被称为极大项，当且仅当它包含变量集 $X$ 的所有变量。对任意子句 $C$，$MC(C)$ 表示 $C$ 扩展得到的极大项。给定子句集 $S$，有 $MC(S)=\bigcup_{i=1}^{n}C_i$。

**定义 3**(规则空间) SAT 公式中将每个子句扩展为极大项，每个极大项视为 SAT 公式的一条规则，所有无重复的极大项组成的集合称为 SAT 公式的规则空间。

**定义 4**(解空间) 每个使得 SAT 公式满足的赋值称为一个解，解的集合称为解空间。给定子句集 $S$，$M(S)$ 表示 $S$ 的解空间。

**定义 5**(互补因子)[7] 互补因子(Complementary Factor)是含有互补对的子句对的个数与所有子句对的个数的比值，即 $M/(n\times(n-1)/2)$，$M$ 表示含有互补对的子句对的个数。通常子句集的互补因子越大，子句两两之间互补的可能性就越大，基于扩展规则的 ♯SAT 算法效率也越高。

## 3 知识编译

知识编译(Knowledge Compilation)起源于 20 世纪 90 年代[8]，其主要目标是为那些需要大量内存或时间来求解的问题提供高效的表示方法。该技术是一种有效处理知识的方法，首先使用命题逻辑表示知识并建立知识库，然后对知识库进行知识编译，以便在查询时能够快速响应。目前的知识编译方法主要处理命题逻辑表示的支持库，并涌现出多种编译目标语言，如 BDD[9] 和 DNNF[10] 等，不同编译目标语言具有各自的性质[11-12]。实质上，知识编译是对知识的处理和转换，主要包括离线推理阶段和在线推理阶段。在离线推理阶段，知识编译算法对给定的知识库进行处理和转换，得到易于推理的表示形式。此处理过程只需进行一次，不依赖于具体查询。然后，在在线推理阶段，利用离线推理阶段处理得到的结果，回答是否能将查询结果推出。这种做法大大提高了推理效率，避免了重复计算。因此，知识编译为解决复杂问题提供了一种高效且便捷的方法，且在人工智能领域有广泛应用。

BDD 是长度为 $n$ 的二进制字符串集合的表示形式，通常等价地认为它是一个二进制值函数，如果长度为 $n$ 的二进制字符串在集合中，则将其映射为 1，如果不在集合中，则映射为 0。在结构上，BDD 是一个有向无环图，有 0 和 1 两个终端节点，其中每个非叶节点有两个后继节点：一个 0 后继节点和一个 1 后继节点。与二叉树一样，要确定字符串是否在由 BDD 表示的集合中，需要从根节点开始，沿着当前节点的 0 或 1 后继进行遍历。当结束于 1 时，表示字符串在集合中；结束于 0 时，表示它不在集合中。在实际问题中存在许多相互共享变量的 PB 约束，将它们全部表示为单个 BDD 能生成更紧凑的 SAT 编码，而且具有更好的传播特性，因此本文选择 BDD 作为编译目标语言。图 1 给出了子句集 $S=(x_1\vee x_2)\wedge(x_1\vee\neg x_2\vee x_3)\wedge(\neg x_1\vee x_2\vee x_3)$ 对应的 BDD 结构。

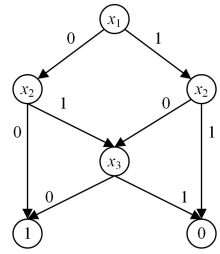



图 1 $S$ 对应的 BDD 结构

Fig. 1 Corresponding BDD structure of $S$

## 4 扩展规则与解空间

扩展规则是由 Lin 等提出的归结的互补方法，通过寻找子句的极大项来判断 CNF 公式的可满足性。对于一个变量集 $M=\{x_1, x_2, \cdots, x_m\}$ 上的子句集 $S=\{C_1, C_2, \cdots, C_n\}$，其中任何一个子句都可以应用扩展规则转化为若干个 $M$ 上的极大项。集合 $M$ 上的 SAT 公式 $F$ 共有 $2^m$ 种赋值方式，每种赋值都是 $M$ 上的极大项。

对于公式 $F$，有：

(1)规则空间大小＋解空间大小＝$2^m$，即 $|MC(S)|+|M(S)|=2^m$；

(2)$F$ 中每新增一条子句，通过扩展规则产生 $x$ 个新的极大项，则 $F$ 的解空间大小减少 $x$；

(3)从 $F$ 的所有赋值方式组成的集合去除规则空间中存在的子句，对剩下的每个子句取否定后都是 $F$ 的一个可满足赋值。

例如：子句集 $S=\{x_1\vee x_3, x_2\vee\neg x_3, x_1\vee\neg x_2\}$，它的规则空间为 $\{x_1\vee x_2\vee x_3, x_1\vee x_2\vee\neg x_3, x_1\vee\neg x_2\vee x_3, x_1\vee\neg x_2\vee\neg x_3, \neg x_1\vee x_2\vee\neg x_3\}$，所有可能取值组成的极大项中 $\{C_1=\neg x_1\vee x_2\vee x_3, C_2=\neg x_1\vee\neg x_2\vee x_3, C_3=\neg x_1\vee\neg x_2\vee\neg x_3\}$ 不出现在规则空间中，$\neg C_1=x_1\wedge\neg x_2\wedge\neg x_3$，$\neg C_2=x_1\wedge x_2\wedge\neg x_3$，$\neg C_3=x_1\wedge x_2\wedge x_3$，即 $x_1=1, x_2=x_3=0$；$x_1=x_2=1, x_3=0$；$x_1=x_2=x_3=1$ 是 $S$ 的解。

区别于求 SAT 问题的具体解，模型计数问题并不需要把所有的极大项都扩展出来，而是只需要计算出极大项的数目。假设 $P_1, P_2, \cdots, P_n$ 分别表示子句 $C_1, C_2, \cdots, C_n$ 通过扩展规则得到的极大项集合，$W$ 是子句集 $S$ 通过扩展规则得到一个无重复的极大项集合，则集合 $W$ 中的元素个数 $|W|$ 可以通过容斥原理求得。

$$\begin{aligned}|W| &= |P_1\cup P_2\cup\cdots\cup P_n| \\ &= \sum_{i=1}^{n}|P_i| - \sum_{1\leqslant i\leqslant j\leqslant n}|P_i\cap P_j| + \cdots + \\ &\quad (-1)^{n+1}|P_1\cap\cdots\cap P_n|\end{aligned}$$

从而可以通过比较 $|W|$ 和公式 $F$ 的解空间大小 $2^m$ 来判断公式 $F$ 的可满足性：当 $|W|=2^m$ 时，公式 $F$ 是不可满足的；当 $|W|\leqslant 2^m$ 时，$F$ 是可满足的，且可满足赋值个数＝$2^m-|W|$。

## 5 模型计数

算法的主要思想是为 PB 约束找到 SAT 编码。即给定一个 PB 约束 $C$，构造与其等价的子句集 $S$，使得任何 $S$ 的可满足解均为 $C$ 的模型。以 PB 约束 $2x_1+3x_2+5x_3\leqslant 6$ 为例，假设变量顺序为 $x_1<x_2<x_3$，以 $x_1$ 为根节点，它的左子树表示

令 $x_1=0$ 得到的 PB 约束 $3x_2+5x_3\leqslant 6$，右子树表示令 $x_1=1$ 得到的 PB 约束 $2+3x_2+5x_3\leqslant 6$，再以 $x_2$ 为子树的根结点重复上述过程，直到遍历完序列中的最后一个变量。当一条分支上的所有变量的取值满足 PB 约束时，将该分支连向真节点“1”；否则将该分支连向假节点“0”。PB 约束 $C=2x_1+2x_2+4x_3\leqslant 5$ 对应的 BDD 如图 2 所示。

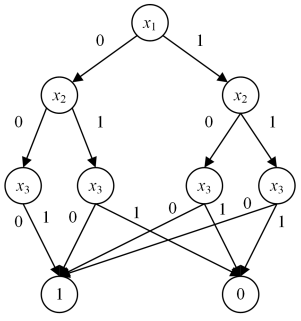


图 2　$C$ 对应的 BDD 结构

Fig. 2　Corresponding BDD structure of $C$

通过删除具有相同子节点的节点和合并同构子树得到图 3 所示的简化 BDD 结构。

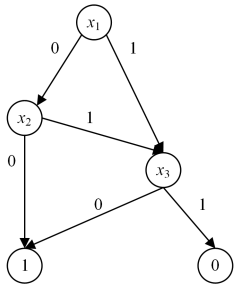


图 3　简化后的 BDD 结构

Fig. 3　Simplified BDD structure

通常在 PB 的具体求解中会使用简化后的 BDD 作为目标结构，因为更少的节点数意味着需要对 BDD 遍历的次数也更少，并且在构造 BDD 时选择的变量顺序也会影响到遍历次数。当变量顺序为 $x_3<x_2<x_1$ 时，对应的 BDD 结构如图 4 所示。图 3 中从根节点到叶子节点共需遍历 5 次，而在图 4 中只需遍历 4 次。

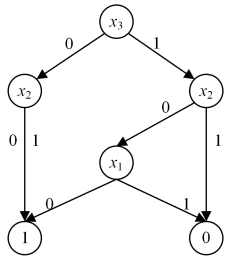


图 4　换序后的 BDD 结构

Fig. 4　BDD structure after reordering

但在模型计数问题中我们选择使用未简化的 BDD，因为未简化的 BDD 相对于简化后的 BDD 有更多的分支，分支越多，转化后得到的子句集中子句数目也越多，需要对子句集进行的扩展次数就越少，而且 BDD 构造时选择的变量顺序不会影响到未简化的 BDD 结构遍历次数，也即对于一个 BDD 结构，不同的变量顺序得到的非极大项数目是不固定的，但扩展后的极大项数目总是固定不变的。将 BDD 结构进行转化，得到与 PB 约束等价的子句集 $(\neg x_1\vee\neg x_2\vee\neg x_3)\wedge(\neg x_1\vee x_2\vee\neg x_3)\wedge(x_1\vee\neg x_2\vee\neg x_3)$，根据解空间的性质可以得到 PB 约束 $2x_1+3x_2+5x_3\leqslant 6$ 有 $2^3-3=5$ 个布尔解。

**算法 1**　BDDConvert

输入：Constraint C=$a_1x_1+\cdots+a_nx_n\leqslant K$；Variable set $\{x_1,x_2,\cdots,x_n\}$

输出：BDD structure B

1. node N
2. for i=1 to n do
3.　　N=$x_i$
4. if bool=0
5.　　$B_F$=BDDConvert(i+1，$a_{i+1}x_{i+1}+\cdots+a_nx_n\leqslant K$)
6.　　$B_T\rightarrow B$
7.　　N→rchild=$B_T$
8. else if
9.　　$B_T$=BDDConvert(i+1，$a_{i+1}x_{i+1}+\cdots+a_nx_n\leqslant K-a_i$)
10.　　$B_F\rightarrow B$
11. N→lchild=$B_F$
12. end for
13. return B

该方法同样可以推广到 PB 约束集合的模型计数问题中。如 PB 约束集合 $C=\begin{cases}2x_1+x_3+3x_4\leqslant 4\\ x_2+2x_3\leqslant 1\end{cases}$，通过知识编译得到两个 BDD 结构，如图 5 和图 6 所示。将其转化为子句集 $S_1=(\neg x_1\vee\neg x_3\vee\neg x_4)\wedge(\neg x_1\vee x_3\vee\neg x_4)$，$S_2=(\neg x_2\vee\neg x_3)\wedge(x_2\vee\neg x_3)$。通过扩展规则得到极大项集合 $W=\{\neg x_1\vee\neg x_2\vee\neg x_3\vee\neg x_4,\neg x_1\vee x_2\vee\neg x_3\vee\neg x_4,x_1\vee\neg x_2\vee\neg x_3\vee x_4,x_1\vee\neg x_2\vee\neg x_3\vee\neg x_4,\neg x_1\vee\neg x_2\vee\neg x_3\vee x_4,x_1\vee x_2\vee\neg x_3\vee x_4,x_1\vee x_2\vee\neg x_3\vee\neg x_4,\neg x_1\vee x_2\vee\neg x_3\vee x_4,\neg x_1\vee x_2\vee x_3\vee\neg x_4,\neg x_1\vee\neg x_2\vee x_3\vee\neg x_4\}$，得到 $|W|=10$。因此得到 $PB$ 约束集合 $C$ 的可满足赋值个数有 $2^4-10=6$ 个。

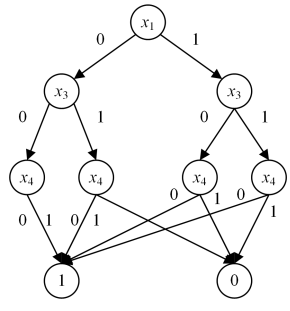


图 5　$S_1$ 对应的 BDD 结构

Fig. 5　Corresponding BDD structure of $S_1$

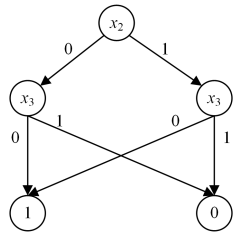


图 6　$S_2$ 对应的 BDD 结构

Fig. 6　Corresponding BDD structure of $S_2$

然而对于变量数和子句数多的子句集，直接利用扩展规则和容斥原理求#SAT问题的空间复杂度是指数级的，因此此处利用Lv等提出的NCER算法[13]对由BDD结构得到的子句集进行模型计数。NCER算法在选择子句进行单文字递归时，利用启发式策略选取当前子句集长度最长的子句，保证子句集规模以最快的速度减小，从而大幅减少算法在求解#SAT问题时递归调用的次数，提高求解效率。

**算法2** NCER

输入：Clause set S；Variable set $X=\{x_1,x_2,\cdots,x_n\}$

输出：Model number

1. if($S=\varnothing$) then
2. return $2^n$
3. if($\varnothing\in S$) then
4. return 0
5. if(unitclause$\{l\}\subset S, x\in l$) then
6. return NCER($\{C-\{\neg l\}\mid C\subset S, l\notin C\}, X-\{x\}$)
7. choose longest length clause $C_i, \{x_1,\cdots,x_k\}\in C_i$
8. $S'=S$
9. for($C'\in S'$)
10. if(C and C′complement each other) then
11. delete C′ from S′
12. else if ($C\neq C'$) then
13. delete the same literal as C in clause C′
14. return $\mathrm{NCER}(S-\{C_i\},X)-\mathrm{NCER}\{S',X-\{x_1,\cdots,x_k\}\}$

## 6 实验与结果

将BDDCovert和NCER算法结合，实现了求解伪布尔公式模型计数的PBM算法。为了验证本文提出的方法PBM的有效性，在3个数据集sensor，knapsack和auctions上验证了其性能，其中每个数据集分别包含1000条数据。

sensor的构建源自关于识别代码集的研究工作[14]。该数据集的主要目标是在给定一个网络图并限定最大传感器数量的条件下，计算出一种放置传感器的方法，以确保网络中的故障是唯一可识别的。在这个背景下，唯一可识别性是指网络中的每个故障都至少能被一组传感器所识别，而且没有其他的故障可以被同一组传感器所识别。

knapsack基于多维背包问题。多维背包问题是一个经典的组合优化问题，最早由Gens和Levner在1980年提出[15]。该问题涉及到一个背包容量有限的场景，其中有$n$个不同的物品，每个物品都具有$m$个不同的特征或维度。这些特征或维度可以代表物品的大小、重量、价值等属性，每个维度都受到一定的约束，例如其总和不得超过给定的常数。在这种情况下，问题的目标是找到一个满足所有约束条件的物品子集，以最大化某个目标函数，通常是子集中物品的总价值或总重量。

auctions的构建基于组合拍卖问题[16]，本文对其进行了调整，使其成为一个计数变体问题。在这个数据集中，考虑有$m$个参与者和$n$个项目的情况，其中每个项目可以由一个或多个参与者共享。假设每个参与者都有一个最小效用阈值，问题的解决目标是计算出一种分配方式，使得这$n$个项目可以被共享的方法的数量最大化，并且同时确保所有的参与者都达到或超过了他们的最小阈值。

为了验证互补因子对模型计数算法的影响，本文对数据集进行了区分，把sensor中数据的互补因子设置为0.6～0.9，knapsack中数据的互补因子设置为0.3～0.6，auctions中数据的互补因子设置为0.1～0.3。

实验中使用的PBM算法是基于C++语言实现的，实验设备采用了CPU主频为3.60 GHz，内存为16 GB，操作系统为Windows 10的微型计算机。

实验部分将PBM算法与最先进的模型计数器DPMC[17]和GPMC[18]进行了比较。由于所有最先进的精确模型计数器都以CNF作为输入，因此本文在使用DPMC和GPMC模型计数器前都先使用了publib[19]对数据集进行转化处理。表1列出了每个模型计数器在数据集上完成实例的具体数目。表2－表4分别列出了PBM算法在3个数据集上的部分实验结果。

表1 各个模型计数器在3600 s内求解的算例数

Table 1 Number of cases solved in 3600 s by each model counter

| 计数器 | sensor | knapsack | auctions | 总数 |
|---|---|---|---|---|
| PBM | 768 | 562 | 278 | 1608 |
| DPMC | 643 | 537 | 485 | 1665 |
| GPMC | 624 | 512 | 507 | 1643 |

表2 PBM在sensor上的部分实验结果

Table 2 Partial experimental results of PBM on sensor

| 算例 | 变量数 | 约束数 | 互补因子 | 布尔解数 | 运行时间/s |
|---|---|---|---|---|---|
| sp12_s43 | 12 | 43 | 0.893 | 1669 | 0.23 |
| sp17_s55 | 17 | 55 | 0.447 | 104648 | 6.28 |
| sp22_s85 | 22 | 85 | 0.625 | 3245607 | 3.47 |
| sp27_s113 | 27 | 113 | 0.384 | 104532756 | 8.93 |
| sp30_s200 | 30 | 200 | 0.253 | 871385471 | 33.65 |

表3 PBM在knapsack上的部分实验结果

Table 3 Partial experimental results of PBM on knapsack

| 算例 | 变量数 | 约束数 | 互补因子 | 布尔解数 | 运行时间/s |
|---|---|---|---|---|---|
| m12_s43 | 12 | 43 | 0.571 | 1787 | 2.3 |
| m17_s55 | 17 | 55 | 0.463 | 129677 | 16.45 |
| m22_s85 | 22 | 85 | 0.426 | 3386482 | 7.82 |
| m27_s113 | 27 | 113 | 0.364 | 117391796 | 25.76 |
| m30_s200 | 30 | 200 | 0.313 | 901327697 | 58.73 |

表4 PBM在auctions上的部分实验结果

Table 4 Partial experimental results of PBM on auctions

| 算例 | 变量数 | 约束数 | 互补因子 | 布尔解数 | 运行时间/s |
|---|---|---|---|---|---|
| a12_s43 | 12 | 43 | 0.291 | 1967 | 13.82 |
| a17_s55 | 17 | 55 | 0.262 | 184762 | 54.48 |
| a22_s85 | 22 | 85 | 0.231 | 3584274 | 57.68 |
| a27_s113 | 27 | 113 | 0.179 | 127239345 | 78.24 |
| a30_s200 | 30 | 200 | 0.127 | 97 6247 331 | 427.50 |

横向来看，在sensor测试集中，PBM计算完成了768个实例，大幅领先于GPMC的624个实例和DPMC的643个实例。在knapsack测试集中，PBM完成了562个实例，略微领先GPMC的512个和DPMC的537个。在auctions测试集中，PBM完成了278个实例，少于GPMC的507个和DPMC的485个。总体而言，PBM完成了3400个实例中的1608个，GPMC完成了1643个，DPMC完成了1665个。虽然PBM完成的总数要比GPMC和DPMC略少，但这一定程度上是因为GPMC和DPMC是直接使用了CNF作为输入，在计算之前都使用了publib处理，先进的预处理能力使得GPMC和DPMC在实例计算之前就具有一定的优势。从分

类后的结果来看，在互补因子高于 0.3 的数据集 sensor 和 knapsack 中，相比 GPMC 和 DPMC，PBM 具有明显的优势。

纵向来看，PBM 算法在 3 组不同的测试集上的计算能力差异非常明显，算例的互补因子越小，PBM 算法需要的运行时间越长。出现这种现象的原因有两个，首先是因为互补因子的大小取决于布尔变元和子句的个数，布尔变元和子句的个数关系影响了子句集的可满足解的个数，也即在互补因子小的测试例子中，PBM 算法需要消耗更多的时间计算出布尔解总数。其次，测试例的互补因子的大小会影响 PBM 生成的 BDD 结构的分支数目，并且在使用扩展规则求极大项数目时需要更多次的扩展操作，从而导致在互补因子低时 PBM 算法的求解效率不佳。

总的来说，PBM 在处理具有高互补因子的情况时表现出更好的求解效果，但在互补因子较低的情况下则效果较差。互补因子的大小对 PBM 算法的性能产生影响的主要原因是直接影响了生成的 BDD 结构的复杂程度。具体而言，较高的互补因子意味着存在更多的互补约束，这会导致生成的 BDD 结构具有更多的分支。然而，BDD 结构中的分支数目直接影响了使用扩展规则算法求解极大项数目时的效率。在存在较多互补约束的情况下，BDD 结构可能变得更加复杂，导致求解极大项的效率降低，从而影响了整体的求解效果。因此，在后续算法优化时需要充分考虑互补因子的大小及其对算法性能的影响。针对不同互补因子的情况，可能需要采用不同的优化策略或调整参数设置，以达到更好的求解效果。这一研究结果对于进一步优化 PBM 算法以及在实际应用中选择合适的求解方法具有重要的指导意义。

**结束语** 本文提出了一种伪布尔约束模型计数方法，用知识编译方法将 PB 约束转化为 BDD 结构，从而利用 BDD 结构的性质将 PB 约束的模型计数问题转化为研究更为成熟的 #SAT 问题，利用基于扩展规则的 NCER 算法对转化得到的极大项集合进行求解。当子句集互补因子较小时，算法求解效果不及基于 CDCL 算法的 PB 约束求解算法；当互补因子较大时，算法求解效果可与传统 PB 约束求解算法相媲美。算法的求解效率主要受互补因子大小的影响，今后将针对互补因子较小的情况研究提升算法求解效率的方法。

## 参 考 文 献


[1] EÉN N, SÖRENSSON N. Translating pseudo-boolean constraints into SAT[J]. Journal on Satisfiability, Boolean Modeling and Computation, 2006, 2(1/2/3/4): 1-26.

[2] ABÍO I, NIEUWENHUIS R, OLIVERAS A, et al. A new look at BDDs for pseudo-Boolean constraints[J]. Journal of Artificial Intelligence Research, 2012, 45: 443-480.

[3] BOFILL M, COLL J, NIGHTINGALE P, et al. SAT encodings for Pseudo-Boolean constraints together with at-most-one constraints[J]. Artificial Intelligence, 2022, 302: 103604.

[4] DING Z Y. Research and Implementation of Applying Linear Algebra to Solve Satisfiability Problem[D]. Guangzhou: Sun Yat-sen University, 2014.

[5] FANG C, LIU J. A linear algebra formulation for boolean satisfiability testing[J]. arXiv: 1701.02401, 2017.

[6] HAI L, JIGUI S, YIMIN Z. Theorem proving based on the extension rule[J]. Journal of Automated Reasoning, 2003, 31: 11-21.

[7] YIN M H, LIN H, SUN J G. Solving #SAT using extension rules[J]. Journal of Software, 2009, 20(7): 1714-1725.

[8] JHA A, SUCIU D. Knowledge compilation meets database theory: compiling queries to decision diagrams[C]// Proceedings of the 14th International Conference on Database Theory. 2011: 162-173.

[9] INOUE K. Evaluating abductive hypotheses using an EM algorithm on BDDs[C]// Proc. IJCAI-09. 2009: 810-815.

[10] DARWICHE A. Decomposable negation normal form[J]. Journal of the ACM(JACM), 2001, 48(4): 608-647.

[11] GU W X. Knowledge Compilation Survey[J]. Computer Science, 2010, 37(7): 7.

[12] ZHAO X W. Research of the Knowledge compilation and Contingent Flexible Planning[D]. Changchun: Northeast Normal University, 2010.

[13] LYU S, ZHANG T, WANG Q, et al. Two Novel Algorithms Based on Extension Rule for Solving # SAT Problem[J]. Journal of Northeastern University(Natural Science), 2019, 40(5): 630.

[14] LATOUR A L D, SEN A, MEEL K S. Solving the identifying code set problem with grouped independent support[J]. arXiv: 2306.15693, 2023.

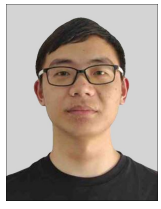
[15] AUSIELLO G, CRESCENZI P, GAMBOSI G, et al. Complexity and approximation: Combinatorial optimization problems and their approximability properties[M]. Springer Science & Business Media, 2012.

[16] ROUGHGARDEN T. Algorithmic game theory[J]. Communications of the ACM, 2010, 53(7): 78-86.

[17] DUDEK J M, PHAN V H N, VARDI M Y. DPMC: weighted model counting by dynamic programming on projectjoin trees[C]// International Conference on Principles and Practice of Constraint Programming. Cham: Springer International Publishing, 2020: 211-230.

[18] SUZUKI R, HASHIMOTO K, SAKAI M. Improvement of Projected Model-Counting Solver with Component Decomposition Using SAT Solving in Components: JSAI Technical Report: SIG-FPAI[R]. 2017: 31-36.

[19] PHILIPP T, STEINKE P. Pblib-a library for encoding pseudo-boolean constraints into cnf[C]// International Conference on Theory and Applications of Satisfiability Testing. Cham: Springer International Publishing, 2015: 9-16.

**ZHENG Suhao**, born in 1999, postgraduate. His main research interests include SAT problem algorithms and theory, and so on.

**NIU Qinzhou**, born in 1956, Ph.D, professor. His main research interests include SAT problem algorithms, and computer network.