arXiv:2301.03667v1 [cs.LO] 9 Jan 2023

阈值综合问题的两种算法实现

Jan-Georg Smaus法国
图卢兹保罗·萨巴捷大学 IRIT

Christian SchillingFabian Wenzelmann德国弗赖堡阿尔伯特·路德维希大学计算机科学研究所

摘要

一个 *linear pseudo-Boolean constraint*（线性伪布尔函数）是如下形式的表达式

$$a_1 \cdot \ell_1 + \ldots + a_m \cdot \ell_m \geq d,$$

其中每个 ℓ_i 都是一个 *literal*（其取值为 1 或 0，取决于命题变量 x_i 为真还是为假），而 $a_1,..,a_m,d$ 是自然数。线性伪布尔函数表示一个布尔函数，而恰好能由一个线性伪布尔函数表示的布尔函数称为 *threshold functions*。若可能，寻找布尔函数的线性伪布尔函数表示这一问题称为 *threshold recognition problem*　*threshold synthesis problem*　$O\ m^7t^5$ 或 。该问题有一种使用线性规划的（ ）算法，其中 m 是维数， t 是析取范式输入中的项数。能否通过完全组合的方法识别阈值函数，一直是个悬而未决的问题。Smaus 开发了这样一种方法：通过分解析取范式并“计数”其中变量的出现次数来完成识别。我们已将两种算法作为毕业论文项目予以实现，并在此报告相关经验。最重要的发现是，遗憾的是，Smaus 的算法并不完备。

1 引言

线性伪布尔约束（LPB）（Dixon 和 Ginsberg 2000；Fr¨anzle 和 Herde 2007）是形如 $a_1\ell_1 + \ldots + a_m\ell_m \geq d$ 的表达式。这里，每个 ℓ_i 都是形如 x_i 或 $\bar{x}_i \equiv 1 - x_i$ 的文字，即当 x_i 为假时， x_i 变为 0；当 x_i 为真时， x_i 变为 1；而 $\bar{x}_i$ 则相反。此外， a_1、.....、 a_m、 d 都是自然数。

线性伪布尔函数可用于表示布尔[1] 函数；例如，$x_1 + \bar{x}_2 + x_3 \geq 3$ 与命题公式 $x_1 \wedge \neg x_2 \wedge x_3$表示同一个函数。研究发现，与合取 或析取 范式（合取范式或析取范式）相比，函数通常可以用一组线性伪布尔函数表示得更紧凑（Dixon and Ginsberg 2000；Fr¨anzle and Herde 2007）。例如，线性伪布尔函数 $2x_1 + \bar{x}_2 + x_3 + x_4 \geq 2$ 对应于析取范式$x_1 \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_4) \vee (x_3 \wedge x_4)$，其中包含四个子句。

[1]每当我们说“函数”时，指的都是“布尔函数”。

本文关注可由单个线性伪布尔函数表示的函数，即所谓的阈值函数。给定一个以析取范式表示的布尔函数，判断它是否为阈值函数，并在可能时计算其线性伪布尔函数表示，这一问题称为阈值识别问题或阈值综合问题。已知可利用线性规划，通过一个 $O(m^7t^5)$算法解决该问题，其中 m 为维度， t为析取范式中的项数（ Crama and Hammer 2011）。

几十年来，能否通过完全组合式的过程识别阈值函数一直是个悬而未决的问题，也就是说，不借助与之等价的线性规划。Smaus 提出了一种方法，通过分解析取范式，并以恰当的方式“计数”其中变量的出现次数来实现（Smaus 2007a）。

弗赖堡大学的学生 Schilling 和 Wenzelmann 分别将经典的线性规划算法和较新的组合算法实现为学士毕业论文项目（Schilling 2011；Wenzelmann 2011）。本文将报告这方面的经验。最重要的发现是，Smaus 的算法遗憾地并不完备。

本文结构如下。接下来介绍一些预备知识。第 3 节介绍线性规划算法，第 4 节介绍组合方法，第 5 节介绍实现，第 6 节总结全文并讨论未来工作。

2 预备知识

我们假设读者熟悉命题逻辑的基本概念。

一个 m**维布尔函数** f 是一个函数 $Bool^m \to Bool$。一个 **线性伪布尔约束** (LPB) 是形如以下形式的不等式

$$a_1\ell_1 + \ldots + a_m\ell_m \geq d \qquad a_i \in \mathbb{N}, d \in \mathbb{Z}, \ell_i \in \{x_i, \bar{x}_i\}. \tag{1}$$

我们称 a_i **系数** 和 d 为**次数** （Hooker 1992）。文字 x_i （或 $\bar{x}_i$）的一次出现称为在 x_i 中的**正** （或**负**）极性出现。注意，如果 $d \leq 0$，则该线性伪布尔函数是重言式。允许负 d 的原因将在第 4.2 小节中显现。

析取范式 是一种形式为 $c_1 \vee \ldots \vee c_n$ 的公式，其中每个**子句** c_j 都是文字的合取。形式上，析取范式是文字集合的集合，也就是说，子句的顺序以及子句内文字的顺序都无关紧要。对于析取范式，不失一般性地假设没有任何子句是另一个子句的子集（后者是多余的，因为它已被吸收）。如果析取范式中的每个子句都是素蕴含项，即对于子句 c_1，不存在子句 $c_2 \neq c_1$ 使得$c_1 \vee c_2 = c_2$，则称该析取范式素且不可约冗余。任何布尔函数都可以表示为析取范式（Wegener 1987）。

容易看出，线性伪布尔函数只能表示单调函数，也就是说，只能表示这样的析取范式：每个变量只以一种极性出现。因此，任何包含以不同极性出现的变量的析取范式都无需考虑。我们不失一般性地假设该极性为正极性。

3 线性规划算法

我们简要概述 Peled & Simeone 提出的线性规划解法（Peled and Simeone 1985；Crama and Hammer 2011）。

对于某些析取范式，可以确定变量上的一个全序 $\succeq$，直观来说，其含义如下：$x_i \succeq x_j$ 当且仅当从任意给定的输入元组 $X^* \in$ 布尔^m 出发，将 x_i^* 设为真，比将 x_j^* 设为真更有可能使该析取范式为真。由此类析取范式表示的函数称为正则函数。

该算法首先检验输入析取范式是否具有正则性。该性质弱于阈值性质，因此，如果某个析取范式不正则，就无法将其转换，我们只能放弃。

通过一种特殊方式统计变量，便可确定顺序。直观而言，如果某个变量出现在许多子句中，且出现在较短子句中，那么它就“很重要”。这一概念被形式化为变量 x 在 ϕ 中的所谓出现模式，记作$OP(\phi, x)$。由于篇幅所限，我们不提供形式化定义，请读者参阅（Smaus 2007a）。

计算 ϕ 中所有变量的出现模式集合，所需时间与 ϕ 的大小成线性关系，因为只需遍历 ϕ 一次即可完成。事实上，所有出现模式中元素的总数恰好等于 ϕ中的文字数。因此，按照出现模式对变量排序所需时间是 $|\phi|$的多项式。

出现模式的概念等价于所谓的*Winder*矩阵（Winder 1962）。我们将在下一节再次用到这个概念。

如果析取范式是正则的，我们就利用其极小真点，即无法将任何一个取值为 1 的变量改为 0 而不使该点变为假点的真元组。我们还使用极大假点，其定义方式与之类似。请注意，这两者共同唯一确定了该析取范式。一般而言，目前尚无已知的多项式算法能够找到这些点（这并不奇怪，因为一般情况下该任务是 NP完全的（Peled and Simeone 1985）），但对于输入析取范式为素不可约形式这一特殊情况，则可以做到。原因在于，可以直接从子句中读取真点。因此，我们要求输入析取范式采用素不可约形式。

有了这些，就存在一个多项式时间过程来求出极大假点。然后我们可以构造如下线性规划，其中极小真点为 x^1、……、x^k，极大假点为 y^1、……、y^l：

$$\begin{aligned} \sum_{i=1}^{m} a_i x_i^j &\geq d \quad (1 \leq j \leq k) \\ \sum_{i=1}^{m} a_i y_i^j &< d \quad (1 \leq j \leq l) \\ a_i &\geq 0 \quad (1 \leq i \leq m) \end{aligned}$$

注意，权重 a_i 是线性规划表述中的变量，阈值为 d。最后，将该线性规划交由线性规划求解器处理。复杂度激增（$O(m^7 t^5)$，其中 m 是维度，t 是析取范式中的项数）主要是由线性规划造成的。其他部分的运行时间为 $O(m^2 t)$，因此整个过程会受益于线性规划未来的改进。值得一提的是，对于大多数输入，用于求解线性规划的著名单纯形法可在线性时间内运行。

4 组合算法

本节回顾我们先前工作（Smaus 2007a）中的结果，并介绍一种算法，用于将析取范式转换为等价的线性伪布尔函数（如果可能的话）。

4.1 确定系数的顺序

给定一个析取范式 ϕ，如果 ϕ 可以表示为线性伪布尔函数，那么其系数必须遵循上一节引入的顺序 $\succeq$，即 $OP(\phi, x_i) \succeq OP(\phi, x_k)$ 意味着所得线性伪布尔函数中的 $a_i \geq a_k$：

引理 4.1 设 ϕ 是由线性伪布尔函数 $\sum_{i=1}^{m} a_i x_i \geq d$ 表示的析取范式。那么 $a_i \geq a_k$ 意味着 $OP(\phi, x_i) \succeq OP(\phi, x_k)$；此外，存在一个线性伪布尔函数 $\sum_{i=1}^{m} a'_i x_i \geq d'$ 表示 ϕ，使得 $OP(\phi, x_i) = OP(\phi, x_k)$ 意味着 $a'_i = a'_k$。

在我们的算法中，有一个概念是对称：如果交换析取范式中的两个变量后得到的仍是同一个析取范式，则这两个变量是对称的。由于篇幅有限，后文将不讨论这一方面，读者可参阅（Smaus 2007a）。

4.2 分解析取范式

我们希望找到一个表示 ϕ 的线性伪布尔函数（LPB），如果可能的话。利用引理 4.1，我们可以确定系数的顺序。假设变量的编号满足 $OP(\phi, x_1) \succeq \ldots \succeq OP(\phi, x_m)$ 这一关系。现在考虑满足 $OP(\phi, x_1) = \ldots = OP(\phi, x_l)$ (=: $OP(\phi, X)$) 的最大集合 $X = \{x_1, \ldots, x_l\}$。（

当然，也完全可能出现 $X = \{x_1\}$，即$l = 1$。）我们希望将 ϕ 划分为若干子问题。为此，我们按照每个子句包含多少个来自 X 的变量，将 ϕ 划分。然后从每个子句中移除来自 X 的变量，由此得到 $l+1$ 个子问题（析取范式）。下文的定理 4.6 说明了这些子问题的解在什么条件下可以组合成表示 ϕ 的线性伪布尔函数（LPB）。然而，由于这些解必须在某种意义上彼此相似，我们不能简单地独立求解各个子问题，然后再将解组合起来，而必须并行求解这些子问题，正如第 4.3 小节将要说明的那样。

以下陈述不要求 X 是极大的。例如，如果 $\{x_1, \ldots, x_5\}$ 是满足$OP(\phi, x_1) = \ldots = OP(\phi, x_5)$ 的极大集合，那么这些陈述对 $X = \{x_1, x_2, x_3\}$ 也成立。从现在起，字母X 总表示上述这类集合，无论它是否极大。

定义 4.2 设 ϕ 是一个析取范式，X 是其变量的一个子集，且满足 $|X| = l$。如果 ϕ 包含一个子句 $c \subseteq X$，那么令 $k_{\max}$ 为此类子句中最长子句的长度；否则令 $k_{\max} := \infty$。对于 $0 \leq k \leq l$，我们将 $S(\phi, X, k)$ 定义为 ϕ 中的子句的析取，这些子句恰好包含$\min\{k, k_{\max}\}$ 个来自 X 的变量，并移除这些变量。

在根据 ϕ 构造$S(\phi, X, k)$ 时，我们称将 X 中的变量从 ϕ 中分离出去。

例 4.3 设 $\phi \equiv (x_1) \vee (x_2) \vee (x_3 \wedge x_4)$ 且 $X = \{x_1, x_2\}$。我们有 $k_{\max} = 1$。于是 $S(\phi, X, 0) = (x_3 \wedge x_4)$，$S(\phi, X, 1)$ = 为真（即两个空合取的析取），且 $S(\phi, X, 2)$ = 为真。

我们必须以一种方式求解这些 $l+1$ 子问题，使所得线性伪布尔约束的所有系数都一致，并且相邻线性伪布尔约束的次数差始终相同。在给出定理之前，我们先通过两个例子加以说明。

例 4.4 考虑 $\phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$ 以及 $X = \{x_1\}$。于是 $S(\phi, X, 0) = x_2 \wedge x_3 \wedge x_4$，可表示为 $x_2 + x_3 + x_4 \geq 3$。此外，$S(\phi, X, 1) = x_2 \vee x_3 \vee x_4$，可表示为 $x_2 + x_3 + x_4 \geq 1$。

由于两个线性伪布尔约束的系数相同，结果表明 ϕ 可以由 $2x_1 + x_2 + x_3 + x_4 \geq 3$表示。$x_1$ 的系数等于两个次数之差，即 $3 - 1$。

例 4.5 考虑 $\phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$ 以及 $X = \{x_1, x_2\}$。我们有 $S(\phi, X, 0) =$为假，由 $x_3 + x_4 \geq 4$ 表示；$S(\phi, X, 1) = x_3 \wedge x_4$，由 $x_3 + x_4 \geq 2$表示；$S(\phi, X, 2) =$为真，由 $x_3 + x_4 \geq 0$表示。析取范式 ϕ 由 $2x_1 + 2x_2 + x_3 + x_4 \geq 4$表示。$x_1, x_2$ 的系数为 $4 - 2 = 2 - 0 = 2$（这些次数“等距”）。

定理 4.6 设 ϕ 是关于变量 $x_1, \ldots, x_m$ 的析取范式，并假设 $X = \{x_1, \ldots, x_l\}$ 是对称变量，使得 $OP(\phi, X)$ 在 ϕ 中关于 $\preceq$ 取最大值。那么 ϕ 可由线性伪布尔函数 $\sum_{i=1}^{m} a_i x_i \geq d$ 表示，其中 $a_1 = \ldots = a_l$，当且仅当对于所有 $k \in [0..l]$，析取范式 $S(\phi, X, k)$ 可由 $\sum_{i=l+1}^{m} a_i x_i \geq d - k \cdot a_1$ 表示。

剩下的问题是，析取范式可能由各种线性伪布尔函数表示，因此即使递归计算出的线性伪布尔约束的系数不一致且次数不等距，也可能找到其他线性伪布尔函数（例如例 4.5 中表示假 的不明显的线性伪布尔函数），从而应用定理 4.6。

在处理这个问题之前，我们通过记录度在不改变含义的情况下可以移动的范围，来推广线性伪布尔约束。为此，我们暂时取消系数和度必须为整数的限制。如何最终得到整数，将在第 4.3 小节末尾说明。

定义 4.7 给定一个线性伪布尔约束 $I \equiv \sum_{i=1}^{m} a_i x_i \geq d$，若 s 是使得对任意 $s' \in (s, d]$，线性伪布尔约束 $\sum_{i=1}^{m} a_i x_i \geq s'$ 都表示与I相同函数的最小数（可能为 $-\infty$），则称 s 为 I 的最小度。若 b 是使得 $\sum_{i=1}^{m} a_i x_i \geq b$ 表示与 I 相同函数的最大数（可能为 ∞），则称 b 为最大度。

注意，I 的最小度本身并不是 I的一个可能度。由于线性伪布尔约束的最小度和最大度比其实际度更有信息量，我们引入记号 $\sum_{i=1}^{m} a_i x_i \geq (s, b]$，用来表示最小度为 s、最大度为 b的线性伪布尔约束。

下面的引理加强了定理 4.6，指出只需很小的额外开销，就能维护有关最小度和最大度的信息。

引理 4.8 作出与定理 *4.6* 相同的假设，并假设对所有 $k \in [0..l]$，析取范式 $S(\phi, X, k)$ 由 $I^k \equiv \sum_{i=l+1}^{m} a_i x_i \geq d - k \cdot a_1$ 表示。此外，对所有 $k \in [0..l]$，令 s_k, b_k 分别为 I^k 的最小度和最大度。那么$s :=$ 最大值$_{k \in [0..l]}(s_k + k \cdot a_1)$, $b :=$ 最小值$_{k \in [0..l]}(b_k + k \cdot a_k)$ 是 $\sum_{i=1}^{m} a_i x_i \geq d$ 的最小度和最大度。

4.3 线性伪布尔约束的组合

定理 4.6 提出了一个递归算法。至少从概念上讲，在基本情形中，我们至多有 2^m 个简单问题需要确定线性伪布尔函数；之所以简单，是因为我们必须为其求线性伪布尔函数的公式要么是真，要么是假。

例 *4.9* 考虑 $\phi \equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_1 \wedge x_5) \vee (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_3 \wedge x_4 \wedge x_5)$。要找到 ϕ 的线性伪布尔函数，我们必须找到 $S(\phi, \{x_1\}, 0)$ 和 $S(\phi, \{x_1\}, 1)$ 的线性伪布尔约束。要找到 $S(\phi, \{x_1\}, 0)$ 的线性伪布尔函数，我们必须找到 $S(S(\phi, \{x_1\}, 0)\ \{x_2\}, 0)$ 和 $S(S(\phi, \{x_1\}, 0)\ \{x_2\}, 1)$ 的线性伪布尔约束，依此类推。表 *1* 给出了所有需要寻找线性伪布尔约束的公式。为简洁起见，我们使用一些缩写，并以右上角的 $S(\cdot, x_{3..5}, 0) \equiv f$ 为例进行说明：它表示 $S((x_3 \wedge x_4 \wedge x_5)\ \{x_3, x_4, x_5\}, 0) \equiv$ *false*，也就是说，“$\cdot$”指左侧最近的非阴影公式，此处为 $(x_3 \wedge x_4 \wedge x_5)$。请注意我们在表中排列子问题公式的方式：例如，$(x_3 \wedge x_4 \wedge x_5)$ 有三个对称变量，拆分这些变量后即可得到待求解的子问题，因此这些子问题

ϕ	$S(\cdot, x_1, 0)$ $\equiv (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_3 \wedge x_4 \wedge x_5)$	$S(\cdot, x_2, 0) \equiv (x_3 \wedge x_4 \wedge x_5)$ $S(\cdot, x_2, 1) \equiv x_3 \vee x_4$	$S(\cdot, x_3, 0) \equiv f$ $S(\cdot, x_3, 1) \equiv (x_4 \wedge x_5)$	$S(\cdot, x_{3..4}, 0) \equiv f$ $S(\cdot, x_{3..4}, 1) \equiv f$ $S(\cdot, x_{3..4}, 2) \equiv x_5$	$S(\cdot, x_{3..5}, 0) \equiv f$ $S(\cdot, x_{3..5}, 1) \equiv f$ $S(\cdot, x_{3..5}, 2) \equiv f$ $S(\cdot, x_{3..5}, 3) \equiv t$
			$S(\cdot, x_3, 0) \equiv x_4$ $S(\cdot, x_3, 1) \equiv t$	$S(\cdot, x_{3..4}, 0) \equiv f$ $S(\cdot, x_{3..4}, 1) \equiv t$ $S(\cdot, x_{3..4}, 2) \equiv t$	
	$S(\cdot, x_1, 1)$ $\equiv x_2 \vee x_3 \vee x_4 \vee x_5$	$S(\cdot, x_2, 0) \equiv x_3 \vee x_4 \vee x_5$ $S(\cdot, x_2, 1) \equiv t$	$S(\cdot, x_{2..3}, 0) \equiv x_4 \vee x_5$ $S(\cdot, x_{2..3}, 1) \equiv t$ $S(\cdot, x_{2..3}, 2) \equiv t$	$S(\cdot, x_{2..4}, 0) \equiv x_5$ $S(\cdot, x_{2..4}, 1) \equiv t$ $S(\cdot, x_{2..4}, 2) \equiv t$ $S(\cdot, x_{2..4}, 3) \equiv t$	$S(\cdot, x_{2..5}, 0) \equiv f$ $S(\cdot, x_{2..5}, 1) \equiv t$ $S(\cdot, x_{2..5}, 2) \equiv t$ $S(\cdot, x_{2..5}, 3) \equiv t$ $S(\cdot, x_{2..5}, 4) \equiv t$

表 1：例 4.9 的递归问题

位于 $x_3 \wedge x_4 \wedge x_5$三 列的右侧()。中间的两个阴影框包含的子问题分别是仅拆分 $\{x_3\}$、 $\{x_3, x_4\}$ 后得到的子问题。还要注意最后一列中的空框，这是因为我们没有尝试将 x_5 与 $x_3 \vee x_4$ 拆分开。

我们提出的算法并非纯粹递归算法，因为每一层的子问题都必须并行求解。以该例说明，我们首先为最右列中的公式求出线性伪布尔约束，这些公式包含 0 个变量，因此我们必须确定 0 个系数。再往左，公式至多包含 x_5，我们为这些公式确定相应的线性伪布尔约束，并且对所有公式都使用相同的 a_5！接下来我们确定 a_4，依此类推。

以表 1 中的 $(x_3 \wedge x_4 \wedge x_5)$ 为例，定理 4.6 表明，a_3， a_4， a_5 应当相等（x_3， x_4， x_5 是对称的），并一次性确定。然而，由于 a_3， a_4， a_5 还必须表示其他子问题公式，而其中的 x_3， x_4， x_5 不一定对称，因此不能一次性确定 a_3， a_4， a_5 ，而应先确定 a_5，再确定 a_4，然后确定 a_3。因此，有必要定义并解释那些按定理 4.6 的含义、分离出的变量少于可分离数量的公式。这些就是阴影公式。

对于每个 $l \in \{0, \dots m\}$，我们将第 $l+1$ 列中的公式称为 l-后继。阴影公式称为辅助公式，其余称为主要公式。右侧没有其他公式的公式称为最终公式。以下定义将这些概念形式化。

定义4.10 设 ϕ 是含有 m 个变量的析取范式。那么 ϕ 是 ϕ 的 0-后继 。此外， ϕ 是 ϕ 的主要后继。而且，如果 ϕ' 是 ϕ 的主要 n-后继，且 l 取最大值，使得 x_{n+1}、……、x_{n+l} 在 ϕ' 中对称，那么对于所有满足 $1 \le l' \le l$ 和 $0 \le k \le l'$ 的 l'、k，我们称 $S(\phi', \{x_{n+1}, \dots, x_{n+l'}\}, k)$ 是 ϕ 的$(n+l')$-后继。$(n+l)$-后继称为主要后继；对于 $l' < l$，$(n+l')$-后继称为辅助后继。一个既是主要节点又为真或假的节点是称为最终。

特别注意表 1 第 3 列中的 $x_3 \vee x_4$。它不包含 x_5，因此我们在倒数第二列中得到最终 4-后继。显然，ϕ 的最终后继要么是真，要么是假。

命题 4.11 假设 ϕ、 ϕ'、 n、 l 如定义 *4.10* 所述。对于 $0 < l' < l$ 和 $0 \le k \le l'$，有

$$S(S(\phi', \{x_{n+1}, \dots, x_{n+l'}\}, k), \{x_{n+l'+1}\}, 0) \equiv S(\phi', \{x_{n+1}, \dots, x_{n+l'+1}\}, k)$$
$$S(S(\phi', \{x_{n+1}, \dots, x_{n+l'}\}, k), \{x_{n+l'+1}\}, 1) \equiv S(\phi', \{x_{n+1}, \dots, x_{n+l'+1}\}, k+1)$$

例如，考虑表 1 中的 $S((x_3 \wedge x_4 \wedge x_5)\ \{x_3\},\ 1) \equiv (x_4 \wedge x_5)$。我们有 $S((x_4 \wedge x_5)\ \{x_4\},\ 0) \equiv S((x_3 \wedge x_4 \wedge x_5)\ \{x_3, x_4\},\ 1)$，以及 $S((x_4 \wedge x_5)\ \{x_4\},\ 1) \equiv S((x_3 \wedge x_4 \wedge x_5)\ \{x_3, x_4\},\ 2)$。一般而言，每个非终结后继都与其紧邻右侧列中的两个公式相关联，一个略高，一个略低；这两个公式是通过拆分出索引最小的变量得到的。

这本身并不令人意外，这对应于一种朴素的方法：我们每次总是拆分出一个变量（以应用定理 4.6），从而构造右侧一列中的公式。命题 4.11 的要点在于，我们通常可以构造更少的公式，因为$S(S(\phi, \{x_{n+1}, \dots, x_{n+l'}\}, k)\ \{x_{n+l'+1}\}, 1)$ 和 $S(S(\phi, \{x_{n+1}, \dots, x_{n+l'}\}, k+1)\ \{x_{n+l'+1}\}, 0)$ 相同。这意味着，ϕ' 会触发 $l+1$ 个主要的 $(n+l)$-后继，而不是 2^l。在表 1 中，我们有 12 个最终公式，而不是 $2^5 = 32$。

一般来说，表中的最终节点似乎比 2^m少得多。我们考察的许多例子强烈表明，即使尝试构造一个对称性尽可能少、因而会导致表格很大的输入析取范式，通过拆分构造出的子公式仍总是具有许多对称性。若能针对这一观察给出一个理论性结论，将会很有意思。

以下定理说明了是否以及如何求出表示所有 k-

$4x_1+3x_2+$ $2x_3+2x_4+$ $x_5\geq\ldots$	$3x_2+$ $2x_3+2x_4+$ $x_5\geq\ldots$	$2x_3+2x_4+$ $x_5\geq\ldots$	$2x_4+$ $x_5\geq\ldots$	$x_5\geq\ldots$	$\sum_{i=6}^{5}a_ix_i$ $\geq\ldots$
$(4,5]$	$(4,5]$	$(4,5]$	$(3,\ \infty]$	$(1,\ \infty]$	$(0,\infty]$
				$(1,\ \infty]$	$(0,\infty]$
			$(2,3]$	$(0,1]$	$(0,\ \infty]$
					$(-\infty,\ 0]$
		$(1,2]$	$(1,2]$	$(1,\ \infty]$	
			$(-\infty,\ 0]$	$(-\infty,\ 0]$	
				$(-\infty,\ 0]$	
	$(0,1]$			$(0,1]$	$(0,\infty]$
		$(0,1]$	$(0,1]$	$(-\infty,\ 0]$	$(-\infty,0]$
		$(-\infty,\ 0]$	$(-\infty,\ 0]$	$(-\infty,\ 0]$	$(-\infty,\ 0]$
			$(-\infty,\ 0]$	$(-\infty,\ 0]$	$(-\infty,\ 0]$
					$(-\infty,\ 0]$

表 2：例 4.9 的线性伪布尔约束

后继所需的下一个系数和次数，前提是已知表示所有 $(k+1)$-后继所需的系数和次数。

定理 4.12 假设 ϕ 如定理 *4.6* 中所述，并且存在某个k 满足 $0\leq k\leq m-1$，令 Φ_k 为 k-后继的集合，其中 ϕ。对于每个非终结的 $\phi'\in\Phi_k$，假设我们有两个线性伪布尔约束 $\sum_{i=k+2}^{m}a_ix_i\geq(s_{\phi'0},b_{\phi'0}]$ 以及 $\sum_{i=k+2}^{m}a_ix_i\geq(s_{\phi'1},b_{\phi'1}]$，分别表示 $S(\phi',\{x_{k+1}\},0)$ 和 $S(\phi',\{x_{k+1}\},1)$。

如果可以选择 a_{k+1} 使得

$$\max_{\phi'\in\Phi_k}(s_{\phi'0}-b_{\phi'1})<a_{k+1}<\min_{\phi'\in\Phi_k}(b_{\phi'0}-s_{\phi'1}),\quad(4)$$

那么，对于所有 $\phi'\in\Phi_k$，线性伪布尔函数 $\sum_{i=k+1}^{m}a_ix_i\geq(s_{\phi'},b_{\phi'}]$表示 ϕ'，其中

$$s_{\phi'}=\max\{s_{\phi'0},s_{\phi'1}+a_{k+1}\},$$
$$b_{\phi'}=\min\{b_{\phi'0},b_{\phi'1}+a_{k+1}\}\ \textit{for non-final}\ \phi';\quad(5)$$
$$s_{\phi'}=-\infty,\ b_{\phi'}=0\ \textit{for}\ \phi'\equiv\textit{true};\quad(6)$$
$$s_{\phi'}=\textstyle\sum_{i=k+1}^{m}a_i,\ b_{\phi'}=\infty\ \textit{for}\ \phi'\equiv\textit{false}.\quad(7)$$

若 $\max^{\phi'\in\Phi_k}(s_{\phi'0}-b_{\phi'1})\geq\min^{\phi'\in\Phi_k}(b_{\phi'0}-s_{\phi'1})$，则不存在 a_{k+1}，$s_{\phi'},b_{\phi'}$ 使得 $\sum_{i=k+1}^{m}a_ix_i\geq(s_{\phi'},b_{\phi'}]$ 对所有 $\phi'\in\Phi_k$ 均表示 ϕ'。

ϕ 的 m-后继，即最右列中的公式，只能是假 或 真。它们由左侧为空和的线性伪布尔函数表示：$\sum_{i=m+1}^{m}a_ix_i\geq(0,\infty]$ 表示 假，$\sum_{i=m+1}^{m}a_ix_i\geq(-\infty,0]$ 表示 真。然后我们使用定理 4.12继续进行，每一步选择任意一个满足条件(4)的 a_{k+1}。

例 4.13 再次考虑例 *4.9*。表 *2* 与表 *1* 严格对应排列，展示了 ϕ 的所有后继对应的线性伪布尔约束。顶行给出线性伪布尔约束的左侧，显然，同一列中的每个线性伪布尔约束都相同。在主表中，我们列出了每个公式的最小次数和最大次数。

第一步中，应用 *(4)* 时，我们必须选择 a_5，使得

$$\max\{0-\infty,0-\infty,0-0,0-0,-\infty-0,-\infty-0,-\infty-0\}<a_5<\min\{\infty-0,\infty-0,\infty--\infty,\infty--\infty,0--\infty,0--\infty,0--\infty\}.$$

选择 $a_5=1$ 即可。第 *5* 列中的最小度和最大度根据 *(5)* 计算；例如，最上方的 $(1,\ \infty]$ 是 (最大$\{0,0+1\}$，最小$\{\infty,\infty+1\}$]。

下一步，我们必须选择 a_4，使得

$$\max\{1-\infty,1-1,1-0,-\infty-0,0-0,-\infty-0,-\infty-0\}<a_4<\min\{\infty-1,\infty-0,\infty--\infty,0--\infty,1--\infty,0--\infty,0--\infty\}.$$

选择 $a_4=2$ 即可。注意，界 $1-0<a_4$ 来自第 *5* 列中间的方框，因此最终来自 $x_3\vee x_4$。我们的算法强制要求 $a_4>a_5$，这对于表示$x_3\vee x_4$的线性伪布尔函数必须成立。

下一步，a_3 也可以选择为任意数 >1，因此我们再次选择 2。下一步，$2<a_2<4$ 必须成立，因此我们选择 $a_2=3$。最后，$3<a_1<5$ 必须成立，因此我们选择 $a_1=4$。我们得到线性伪布尔函数 $4x_1+3x_2+2x_3+2x_4+x_5\geq(4,5]$。

我们已经通过示例了解了算法的工作方式。然而，由于通常 a_{k+1} 的选择并不唯一，人们可能会担心，错误地选择 a_{k+1} 之后，可能会导致定理 4.12 后续无法应用。与 Smaus（2007b）的说法相反，这确实是个问题。我们建议始终选择尽可能小的整数值作为 a_{k+1}，以获得系数较小的线性伪布尔函数。但事实证明，这种策略有时会使算法陷入死胡同。

例 4.14 考虑析取范式

$$\phi\equiv(x_1\wedge x_2)\vee(x_1\wedge x_3)\vee(x_1\wedge x_4\wedge x_5)\,(x_2\wedge x_3\wedge x_4)\vee(x_2\wedge x_3\wedge x_5)\vee(x_2\wedge x_4\wedge x_5)\vee(x_3\wedge x_4\wedge x_5\wedge x_6)$$

			$2x_4 +$ $2x_5 + x_6 \geq \ldots$	$2x_5 + x_6 \geq \ldots$	$x_6 \geq \ldots$	$0 \geq \ldots$
(?, ?]	(?, ?]	(?, ?]	(5, ∞] (4, 5]	(3, ∞] (3, ∞] (2, 3]	(1, ∞] (1, ∞] (1, ∞] (0, 1]	(0, ∞] (0, ∞] (0, ∞] (0, ∞] (−∞, 0]
		(?, ?]	(3, 4] (1, 2]	(3, ∞] (1, 2] (−∞, 0]	(1, ∞] (1, ∞] (−∞, 0] (−∞, 0]	
	(?, ?]	(?, ?] (−∞, 0]	(3, 4] (−∞, 0] (−∞, 0]	(3, ∞] (1, 2]	(1, ∞] (1, ∞] (−∞, 0]	

表3：例4.14的线性伪布尔约束

我们应用该算法生成 ϕ 的所有后继，并为所有递归子问题计算线性伪布尔函数。相应的线性伪布尔函数见表*3*。按照尽可能选择较小系数的策略，我们选择 $a_6 = 1$、$a_5 = 2$、$a_4 = 2$。我们利用第四列中的最小度和最大度来选择系数 a_3。我们必须选择 a_3，使得

$$\max\{5-5,\, 3-2\,,3-0,\, -\infty-0\} < a_3 < \min\{\infty-4,\, 4-1,\, 4--\infty,\, 0--\infty\}$$

也就是说 $3 < a_3 < 3$。当然，这是不可能的。但 ϕ可以用线性伪布尔约束 $9x_1 + 7x_2 + 6x_3 + 4x_4 + 4x_5 + x_6 \geq 15$ 表示。

该算法找到了第四列中所有子问题的解。但我们无法将目前为止选出的系数组合成一个解，以表示第三列中的所有线性伪布尔约束。

另一种做法是，我们可以选择 $a_5 = a_4 = 4$；若如此，便能得到一个合适的线性伪布尔约束。因此，定理 *4.12* 是否适用取决于先前系数的选择。

另一个问题似乎是，a_{k+1} 可能被迫取相邻整数之间的值，此时它本身就不是整数。然而，在这种情况下，可以在继续操作之前，将当前系统中的所有线性伪布尔约束乘以 2（这显然不会改变这些约束的含义），这样就可以将 a_{k+1} 选为整数。

从后继项的构造（见表 1）可以看出，同一列中所有公式的总大小都小于其左侧一列中所有公式的总大小，因此整个表的大小小于 $|\phi| \cdot (m+1)$。由此可以证明，该算法的复杂度关于 ϕ的大小是多项式级的，而 ϕ 本身的大小则可能关于 m呈指数级增长。事实上，这是最有趣的情况，因为在这种情况下，线性伪布尔函数表示可能带来指数级的节省。

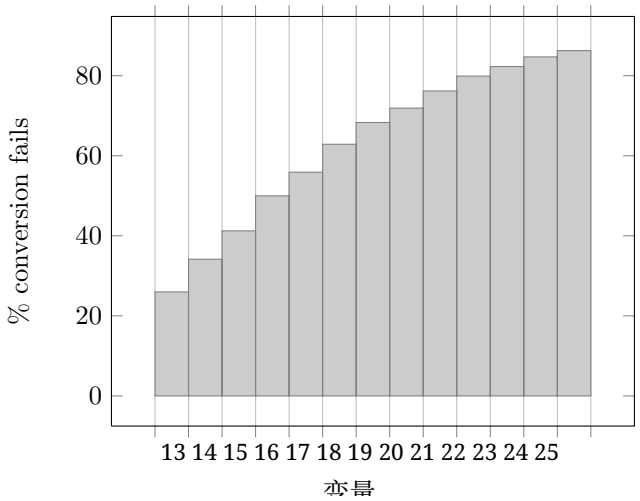


图 1：组合算法的失败率

5 实现

两种算法均已用 Java 实现。它们共用相同的核心类，用于表示析取范式和线性伪布尔约束等主要组成部分。线性规划由 lp_solve 求解。两种实现都可通过图形用户界面访问和测试。

为了测试实现，我们对最多含七个变量的线性伪布尔约束进行了完全枚举。对于含更多变量的线性伪布尔约束，我们测试了 180,000 个随机生成的实例（含 8 至 25 个变量）。我们将这些线性伪布尔约束转换为析取范式（因此我们知道这些析取范式存在对应的线性伪布尔约束），以测试各项实现。正如预期，线性规划算法解决了所有测试过的输入析取范式。

该组合算法能够求解所有变量数不超过五个的输入析取范式。但对于某些含有六个变量的析取范式，该算法会失败（采用尽可能选择

a_{k+1} 尽可能小的策略）。我们的实证分析表明，析取范式包含的变量越多，转换失败的情况就越常见。测试的含七个变量的析取范式中，约有 8% 无法转换；对于含有 25 个变量的析取范式，约有 86% 无法转换。图 1 展示了变量数为 13 至 25 时的失败率。

在线性规划算法的直接运行时间比较中，它的速度更快，但我们仍在努力改进组合算法。

如第 4.3 小节所述，对于组合算法，最终节点 的数量是衡量其理论运行时间的一项重要标准。图2给出了初步印象。很难判断所呈现的增长是否呈指数级，但无论如何，最终节点的数量都远小于 2^m：对于 $m = 25$，约少 50000 倍。

6 结论与未来工作

线性伪布尔约束之所以备受关注，是因为它们通常可以比合取范式或析取范式更紧凑地表示布尔函数，也因为应用于基于合取范式的命题可满足性求解的技术可以推广到线性伪布尔约束（Dixon 和 Ginsberg 2000；Fr¨anzle 和 Herde 2007）。

有些布尔函数可以用单个线性伪布尔约束表示。寻找这种线性伪布尔约束表示的问题称为阈值识别问题。在这项工作中，我们针对该问题实现了两种算法：一种是基于线性规划的经典算法，另一种是我们此前提出的较新算法。最重要的发现是，我们的算法遗憾地并不完备。

当然，未来工作中最重要的课题是尝试重新确立完备性。

实现这一目标的显而易见的方法，是在算法中加入某种回溯机制：如果一个析取范式可以用线性伪布尔函数表示，而我们无法选择a_{k+1}，那是因为我们必定把某个系数 a_{k+2}、……、 a_m 选得太小了，因为到目前为止，我们的策略一直是尽可能选择较小的系数。为了找到解，我们递增系数 a_{k+2}、……、 a_m，并重新评估线性伪布尔约束。我们可以利用最小度和最大度，确保只枚举合法的候选项。我们逐步递增系数，直到能够选择系数 a_{k+1}。

这种方法的一个问题是，对于无法用线性伪布尔函数表示的析取范式，无法保证算法终止，因为下一个系数的取值往往没有上界。不过，我们确信这个问题可以解决，因为应该能够为每个变量推导出某个上界，使得系数没有必要取大于该上界的值（大致来说，系数没有必要比前一个系数大 m 倍以上）。

另一个问题当然是，回溯会使该算法的运行时间变得更糟，而我们非常担心这会破坏该算法的多项式运行时间。

回溯方法已经实现，并且能够为每个经过测试的输入析取范式找到一个解。但实现结果也表明，维度 m 越高，我们就越常需要使用回溯。

或者更可能的是，除了其他方法之外，还可以利用出现模式来估算权重比：在上面的例子中，我们能够表示第四列中的所有线性伪布尔约束，但无法选择 a_3，使其能够在配置 $a_6 = 1$，$a_5 = a_4$ =2 下表示第三列中的所有线性伪布尔约束。我们需要一些全局 信息，以判断后续 a_6 和 a_5，a_4 之间的距离是否会太小。

不过，也许可以利用出现模式来表述此类约束，即可以找到一种如下形式的约束：“在表示 ϕ 的线性伪布尔函数中，必须确保 $a_i \geq w \cdot a_j$”。

不过需要指出的是，过去也有人尝试直接将出现模式转化为数值系数，或者更确切地说，转化为系数比；但阈值识别问题始终难以通过这类尝试解决[2]。

然而，即便只是根据出现模式对系数比进行粗略估算，也可能有助于减少甚至消除回溯工作量。

另一个有趣的课题是更深入地分析组合算法的复杂度，无论是针对其当前状态，还是在使其达到完备之后。尤其是，正如我们在第 4.3 小节中提到的，分析利用输入析取范式中的对称性所产生的影响会很有意思。

致谢我们感谢 Yves Crama 和 Utz-Uwe Haus，他们就这项工作与我们进行了非常有益的讨论。

参考文献

[Crama 和 Hammer 2011] Crama, Y., and Hammer, P. L. 2011. 布尔函数：理论、算法与应用，《数学及其应用百科全书》第142卷。剑桥大学出版社。

[Dixon 和 Ginsberg 2000] Dixon, H. E., and Ginsberg, M. L. 2000. 融合来自人工智能和运筹学的可满足性技术。《知识工程评论》15(1):31– 45.

[Fr¨anzle 和 Herde 2007] Fr¨anzle, M., and Herde, C. 2007. HySAT：一种用于混合系统有界模型检测的高效证明引擎。形式化方法与系统设计30(3):179–198.

[Hooker 1992] Hooker, J. N. 1992. 0-1线性不等式的广义消解。数学、人工智能年刊6(1-3):271–286。

[2]与 Yves Crama 的个人通信

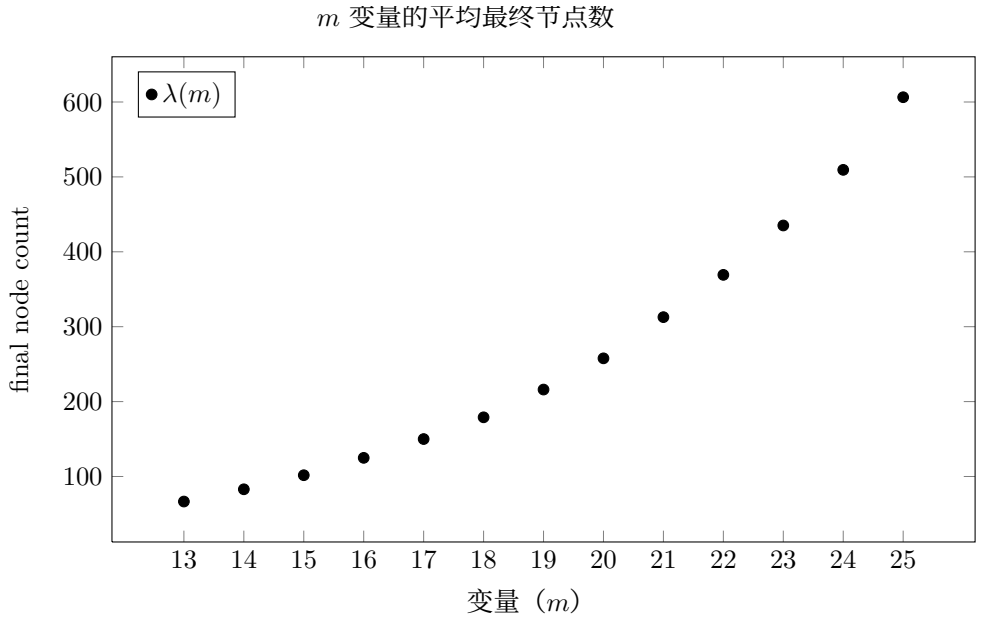


图2： λ（m）是含 m 个变量的析取范式的平均最终节点数。

[Peled 和 Simeone 1985] Peled, U. N., and Si meone, B. 1985. 正则集合覆盖与阈值综合的多项式时间算法。离散应用数学12(1):57–69.[Schilling 2011] Schilling, C. 2011. 通过转换为线性规划求解布尔函数的阈值综合问题。学士论文，弗赖堡阿尔伯特-路德维希大学。[Smaus 2007a] Smaus, J. 2007a. 关于可编码为单个线性伪布尔约束的布尔函数。载于*CPAIOR*，计算机科学讲义第4510卷，288–302页。施普林格。[Smaus 2007b] Smaus, J. 2007b. 关于可编码为单个线性伪布尔约束的布尔函数。技术报告230，弗赖堡大学计算机科学研究所。是（Smaus 2007a）的扩展版本。亦可在 www.avacs.org 获取，编号为技术报告第13号。[Wegener 1987] Wegener, I. 1987. 布尔函数的复杂性。Wiley-Teubner。[Wenzelmann 2011] Wenzelmann, F. 2011. 用组合算法求解布尔函数的阈值综合问题。学士论文，弗赖堡阿尔伯特-路德维希大学。[Winder 1962] Winder, R. O. 1962. 阈值逻辑。博士论文，普林斯顿大学数学系，普林斯顿，美国。