arXiv:2512.21448v1 [cs.LO] 24 Dec 2025

关于通过一阶投影证明划分问题 NP困难性的注记

Pa´ul Risco Iturralde

独立研究者

摘要

在《论计算电路复杂度的（非）NP困难性》一文中，Murray 和 Williams 暗示，尚不清楚划分问题是否可通过 $2^{n^{o(1)}}$-规模的 $\mathbf{AC^0}$ 归约证明为 NP难问题。在本文中，我们证明划分问题可通过一阶投影证明为 NP难问题。基本上，我们对从 3SAT 到子集和以及从子集和到划分问题的众所周知的归约进行了轻微修改，但将其置于描述性计算复杂性的背景下，即使用一阶逻辑公式对这些归约进行定义。由于一阶归约是 $\mathbf{AC^0}$ 归约的一种特定类型，因此可推出在多项式规模 $\mathbf{AC^0}$ 归约下的困难性。由此，本文填补了文献中的一个空白。

1 引言

在其论文 [MW17], Murray 和 Williams 中，他们提到：“TIME $\left(n^{o(1)}\right)$) 归约对于几乎所有具有‘局部’结构的 NP 完全性结果来说都足够强大；这类结构会将输入的小片段转换为输出的小片段”，而从 SUBSET-SUM 到划分问题的典型归约可能是一个反例（其论文中的定义 1.1 解释了什么是 TIME ($t(n)$)) 归约）。在脚注2中，他们提到，从 SUBSET-SUM 到划分问题的直接归约似乎无法在 $2^{n^{o(1)}}$ 大小的 $\mathbf{AC^0}$ 中计算，因为该归约输出中的两个数需要对输入实例中的所有数求和。

在本文中，我们通过一阶投影（fops）证明划分问题是 NP难的；一阶投影是由受限一阶逻辑公式定义的归约。这意味着，在多项式大小的 $\mathbf{AC^0}$ 归约 [ABI97] [CSV84] 下，划分问题仍是 NP难的。

我们如何克服 Murray 和 Williams 提到的障碍？首先，让我们回顾一下子集和、划分问题以及归约本身的定义 [GJ79]。

子集和：
实例： 有限集 B，大小为 $s(a) \in \mathbb{Z}^+$ 对于每个 $a \in B$，以及正整数 t。

问题： 是否存在一个子集 $B' \subseteq B$，使得B' 中元素大小之和恰好为 t?

划分问题：
实例： 一个有限集 A，以及对每个 $a \in A$，一个大小为 $s(a)$ 的 $\in \mathbb{Z}^+$。

问题： 是否存在一个子集 $A' \subseteq A$，使得

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

该归约以一个子集和实例为输入，并输出集合 $A = B \cup \{b_1, b_2\}$，其中

$$s(b_1) = 2\left[\sum_{a \in A} s(a)\right] - t \text{ and}$$
$$s(b_2) = \left[\sum_{a \in A} s(a)\right] + t.$$

现在，一些观察表明，$s(b_1)$ 和 $s(b_2)$ 虽然肯定取决于输入的大小，但实际上可以设法使其不依赖于 B中各元素的大小，方法如下（作者最先观察到，划分问题可通过多项式大小的 AC^0 归约证明为 NP难，见 [Pa´u]）。

在 [Ste94], Stewart 展示了如何通过 fops 证明 3SAT 是 NP 完全，其中每个实例的每个子句都有三个不同文字。Sipser 在其著作 [Sip13], 中介绍了一个从 3SAT 到子集和的（标准）归约。该归约本质上是一个投影，但假设一个文字可以在一个子句中出现多次。尽管如此，若假设每个子句都有三个不同文字，同一归约仍然适用。

现在，使用 Stewart 论文中的 3SAT 术语，每个 3SAT 实例 ϕ 都有n 个子句 $c_0, c_1, \ldots, c_{n-1}$ 以及 n 个变量 $x_0, x_1, \ldots, x_{n-1}$。Sipser 的归约构造了一个子集和实例，其中 B 由以下部分组成

$$y_0, z_0, y_1, z_1, \ldots, y_{n-1}, z_{n-1} \quad \text{and} \quad g_0, h_0, g_1, h_1, \ldots, g_{n-1}, h_{n-1}.$$

相应大小的十进制表示以及目标 t，类似于表 1 中给出的形式。

对于每个 $j \in \{0, 1, \ldots, n-1\}$，对应于行列对 (y_j, j)、(z_j, j)、(g_j, c_j) 和 (h_j, c_j) 的条目值均为 1。此外，对于 $i, j \in \{0, 1, \ldots, n-1\}$，如果文字 x_j 出现在 c_i 中，则 (y_j, c_i) 条目的值为 1；如果文字 $\neg x_j$ 出现在 c_i 中，则 (z_j, c_i) 条目的值为 1。其余条目的值均为 0。还要注意，目标 t 不依赖于 B中各成员的大小。

	0	1	2	3	...	$n-1$	c_0	c_1	c_2	...	c_{n-1}
y_0	1	0	0	0	...	0	1	0	1	...	1
z_0	1	0	0	0	...	0	0	1	1	...	0
y_1		1	0	0	...	0	1	0	0	...	1
z_1		1	0	0	...	0	1	1	0	...	0
y_2			1	0	...	0	0	1	0	...	1
z_2			1	0	...	0	0	0	1	...	0
$\vdots$					$\ddots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$
y_{n-1}						1	0	0	0	...	0
z_{n-1}						1	0	0	0	...	0
g_0							1	0	0	...	0
h_0							1	0	0	...	0
g_1								1	0	...	0
h_1								1	0	...	0
$\vdots$										$\ddots$	$\vdots$
g_{n-1}											1
h_{n-1}											1
t	1	1	1	1	...	1	3	3	3	...	3

表 1：Sipser 的归约中的子集和实例

现在，注意到

$$\sum_{a\in B} s(a) = 2222\ldots2555\ldots5,$$

因为 ϕ 中的每个子句都包含三个不同文字。因此，

$$2\left[\sum_{a\in B} s(a)\right] - t = 3333\ldots3777\ldots7 \text{ and}$$

$$\left[\sum_{a\in B} s(a)\right] + t = 3333\ldots3888\ldots8,$$

也就是说，$s(b_1)$ 和 $s(b_2)$ 将不取决于 B 中各成员的大小，前提是我们假设每个子集和实例都具有表中给出的形式；但如果我们将该问题视为一个限制问题，就可以作出这样的假设。

下一节将介绍理解本文所需的概念，最后一节将展示我们的结果。

2 预备知识

我们假设读者熟悉数理逻辑和计算复杂性理论的基本概念。此外，我们假设读者了解初等电路复杂性。

初等电路复杂性。

接下来，我们将介绍描述复杂性的基础知识。需要更多细节的读者可参阅 [Imm99]。

在描述复杂性中，我们运用数理逻辑中的工具来研究概念，并在计算复杂性理论中建立结果。例如，NP 中的任意问题都可以用一个二阶存在逻辑句来描述，也就是一个逻辑句$\Phi \in \mathrm{SO}\exists$。例如，考虑 SAT 判定问题。为了用逻辑句定义这个问题，我们首先需要一个词汇表。令 $< P^2, N^2 >$为一个由二元关系符号 P^2和 N^2组成的元组。我们通过结构 $\mathcal{A} =< |\mathcal{A}|, P^{\mathcal{A}}, N^{\mathcal{A}} >$来解释这样的词汇表，其中 $|\mathcal{A}|$（称为结构的论域 $\mathcal{A}$）是自然数的一个初始段，$\{0, 1, \ldots, n-1\}$，其中 $n \geq 2$，且 $P^{\mathcal{A}}$和$N^{\mathcal{A}}$ 是该集合上的二元关系，使得

$$\begin{aligned}(j,i) \in P^{\mathcal{A}} \quad &\text{if and only if} \quad \text{literal}\, x_j \,\text{occurs in clause}\, c_i, \text{ and} \\ (j,i) \in N^{\mathcal{A}} \quad &\text{if and only if} \quad \text{literal}\, \neg x_j \,\text{occurs in clause}\, c_i.\end{aligned}$$

这里，x_j 和 c_i 分别是 φ（SAT 实例）中的变量和子句。

我们将 $|\mathcal{A}| = \{0, 1, \ldots, n-1\}$的基数称为结构的大小 $\mathcal{A}$。

根据上一段，结构 $\mathcal{A}$ 可以看作 SAT 问题的一个实例，即 φ，其中包含 n 个变量和 n个子句。因此，如果我们想表示如下公式

$$\varphi \quad \equiv \quad (\neg x_1 \vee x_0 \vee x_2) \wedge (\neg x_2 \vee x_0)$$

在这种设定下，我们会选取 $n = 3$ 并声明

$$\begin{aligned}P^{\mathcal{A}} &= \{(0,0), (2,0), (0,1), (2,2)\} \quad \text{and} \\ N^{\mathcal{A}} &= \{1,0), (2,1), (2,2)\}\,.\end{aligned}$$

注意，$(2, 2) \in N^{\mathcal{A}}$，$P^{\mathcal{A}}$，因为我们不希望出现空子句。

我们将诸如$(j, i) \in P^{\mathcal{A}}$这样的表达式简写为 $P(j, i)$。

现在，使用这个词汇表，定义 SAT 的公式为

$$\Phi_{SAT} \quad \equiv \quad (\exists S)(\forall x)(\exists y)((y \in S \wedge P(y,x)) \vee (y \notin S \wedge N(y,x)));$$

也就是说，$\mathcal{A} \models \Phi_{SAT}$ 当且仅当 $\mathcal{A}$ 是 SAT 的一个是实例。符号 S 表示对公式中变量的一种真值赋值：当且仅当 j 属于 S 所表示的集合时，x_j 被赋值为 1。注意，$\Phi_{SAT} \in \mathrm{SO}\exists$，因为它是一个二阶句子，其中关系变量由存在量词量化。

如果词汇表中包含一个二元输入关系符号，比如 W^2，我们可以将$W^{\mathcal{A}}$视为一个二元矩阵，其中 (j, i) 项为 1，当且仅当 $(j, i) \in W^{\mathcal{A}}$。当我们使用诸如“ W 矩阵”这样的术语时，指的就是这个矩阵。

在描述复杂性中，我们将判定问题称为布尔查询。更具体地说，布尔查询是与某个问题的所有是实例相对应的结构集合。具有给定词汇表 τ 的所有结构的集合记作 STRUC[τ]。

尽管我们将在本文中考虑的词汇表不包含以下符号，但它们同样属于这些词汇表： $=$、$\leq$、SUC、PLUS、TIMES 0、1， max。最后三个被称为数值常量符号，而其他符号被称为数值关系符号。另一方面，输入关系符号和输入常量符号指的是不在前述列表中的符号，因此，例如，上面的 P 和 N 就是输入关系符号。

数值公式是指其中不出现任何输入关系符号的公式。

现在，我们给出几个定义：

$$\begin{aligned} \mathrm{SUC}(i,j) \quad & \text{if and only if} \quad j \text{ is the succesor of } i \text{ and} \\ \mathrm{PLUS}(i,j,k) \quad & \text{if and only if} \quad i+j=k. \end{aligned}$$

数值常量符号 0、1 和 max 将分别解释为数字 0、1 和 $n-1$。最后， $=$ 和 $\leq$ 将采用其通常的含义。

现在，描述性语境中的标准归约概念是一阶归约，即可以由一阶逻辑公式定义的多对一归约。

令 σ 和

$$\tau =< R_1^{a_1}, R_2^{a_2}, \ldots, R_r^{a_r}, c_1, c_2, \ldots, c_s >$$

为两个词汇表，其中 $R_i^{a_i}$是元数为 a_i的输入关系符号，而c_j 是输入常量符号。如果 S 和 T 是布尔查询，其底层词汇表分别为 σ 和 τ，则从 S 到 T 的 k-元一阶归约是一个函数 $I : \mathrm{STRUC}[\sigma] \rightarrow \mathrm{STRUC}[\tau]$ ，由一组词汇表为σ的一阶公式定义，

$$< \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_r, \psi_1, \psi_2, \ldots, \psi_s >,$$

使得对于所有 $\mathcal{A} \in$STRUC[σ],

$$\mathcal{A} \in S \quad \text{if and only if} \quad I(\mathcal{A}) \in T,$$

其中

$$I(\mathcal{A}) =< |I(\mathcal{A})|, R_1^{I(\mathcal{A})}, R_2^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})}, c_1^{I(\mathcal{A})}, c_2^{I(\mathcal{A})}, \ldots, c_s^{I(\mathcal{A})} >$$

由下式给出：

$$|I(\mathcal{A})| = \left\{ (b_1, \ldots, b_k) \in |\mathcal{A}|^k : \mathcal{A} \models \varphi_0 (b_1, \ldots, b_k) \right\},$$

$$R_i^{I(\mathcal{A})} = \{ (d_1, \ldots, d_{a_i}) \in |I(\mathcal{A})|^{a_i} : \mathcal{A} \models \varphi_i (d_1, \ldots, d_{a_i}) \}$$

对于 $i = 1, \ldots, r$，并且存在唯一的 $(b_1, \ldots, b_k) \in |I(\mathcal{A})|$，使得$\mathcal{A} \models \psi_j(b_1, \ldots, b_k)$，此结论对 $j = 1, \ldots, s$ 成立。

现在，一阶投影是一种一阶归约，其中公式 φ_i（对 $i = 1,2, \ldots, r$）以及公式 ψ_j 具有以下形式

$$\alpha_0 \vee (\alpha_1 \wedge \lambda_1) \vee (\alpha_2 \wedge \lambda_2) \ldots (\alpha_s \wedge \lambda_s),$$

其中 α_e 是数值公式，且两两互斥； λ_e 是文字，其底层关系符号是输入关系符号。此外， φ_0, 定义 $|I(\mathcal{A})|$ 的公式是数值公式。

如果存在一个一阶投影，将布尔查询 S 归约为布尔查询 T, ，我们记作 $S \leq_{fop} T$。 $\leq_{fop}$ 关系具有传递性。

正如 [ABI97]，中所指出的，基于一阶归约的 NP困难性意味着该问题不属于 AC^0 类。此外，根据 Razborov-Smolensky 定理，这种意义下的 NP困难性保证问题不存在任何素数 p 对应的 $AC0[p]$ 电路（否则，多数函数将具有此类电路）。

3 结果

我们给出归约，证明 3SAT $\leq_{fop}$ 子集和以及子集和 $\leq_{fop}$ 划分问题。由于 3SAT（其中每个实例的每个子句都含有三个不同文字）可通过 fops 证明为 NP难，因此即可得出该结果。我们的归约之所以有效，基本上是因为我们在标准归约中已有的 1 之间添加了若干列和行的 0（参见表 1），因此非零列中各项的和保持不变。

Stewart 论文中使用的 3SAT 术语为 $< P^2, N^2 >$。当且仅当变量 x_j 出现在子句 c_i中时， $P(j, i)$ 成立；当且仅当文字 $\neg x_j$ 出现在 c_i中时， $N(j, i)$ 成立。这里，假设每个实例中的每个子句都包含三个不同文字。

另一方面，我们使用的子集和词汇表为 $< W^2, L^1 >$。当且仅当 $s(i)$ 的第 j位为 1 时， $W(i, j)$ 成立；这里，最低有效位位于第 $(n-1)$ 位（假设该结构的论域大小为 n）。 $L(j)$ 表示目标 t 的第j位为 1。

按照我们对 W 的解释，与之关联的二元矩阵的第 i 行，恰好是元素 i, $s(i)$ 的大小，以二进制表示。

我们将定义的有限元运算 ρ_1 是四元的。首先，我们概述归约过程，然后给出相应的公式。如果输入结构的论域大小为 n，那么相应输出结构的论域大小将为 n^4。更准确地说，论域中的元素将是四元组，其中每个分量都是 $\{0, 1, \ldots, n-1\}$ 的一个元素。输出结构所附带的序是字典序。

现在，如果 j 是表 1 前半部分中的一列，我们就给它赋予标签 $(0, j, 1, n-1)$。行 y_j 和 z_j 将分别标记为 $(0, j, 1, n-2)$ 和 $(0, j, 1, n-1)$。接下来，列 c_i 将对应于 $(n-1, i, 1, n-1)$，行 g_i 和 h_i 则分别对应于 $(n-1, i, 1, n-2)$ 和 $(n-1, i, 1, n-1)$。列 j 与行 z_j，以及列 c_i 与行 h_i 具有相同标签这一事实，将是我们第二次归约的关键。

有了这些标签，就可以轻松定义与行列对 $(y_j,\ j)$ 和 $(z_j,\ j)$ 对应的 1，以及与 $(g_i,\ c_i)$ 和 $(h_i,\ c_i)$ 对应的 1。其余的 1 则通过使用 $P(j,\ i)$ 和 $N(j,\ i)$ 以及这些标签来定义。

不难验证，对所有 $n \geq 2$，非零列之间至少有三列全为 0。请记住，我们对四元组使用字典序。因此，例如，连续的非零列 $(n-1, i-1, 1, n-1)$ 和 $(n-1, i, 1, n-1)$ 之间，有带标签的列 $(n-1, i, 0, 0)$、$(n-1, i, 0, 1)$ 和 $(n-1, i, 1, 0)$，这些列全为 0。这一点很重要，因为我们希望将 $t = 1111\ldots1333\ldots3$ 中的 3 写成二进制；也就是说，写出目标时，我们希望在列 $(n-1, i, 1, n-2)$（该列全为 0）和 $(n-1, i, 1, n-1)$ 中出现 1。

让我们定义 ρ_1。

φ_0 ≡**真**。
$\varphi_1(x_1、\ x_2、\ x_3、\ x_4、\ w_1、\ w_2、\ w_3、\ w_4)$ 为每个具有词汇表 $< P^2、\ N^2 >$ 的结构 $\mathcal{A}$ 定义 $W^{\rho_1}(\mathcal{A})$。
行列对中的 1（$y_j、\ j$）（即 $((0、\ j、\ 1、\ n-2)、(0、\ j、\ 1、\ n-1))$ 行列对）：

$$\begin{aligned}\alpha_{00} \equiv\ & x_1 = 0 \wedge x_3 = 1 \wedge \mathrm{SUC}(x_4, max) \quad \wedge \\ & w_1 = 0 \wedge w_2 = x_2 \wedge w_3 = 1 \wedge w_4 = max\end{aligned}$$

行列对中的 1（$z_j、\ j$）：

$$\begin{aligned}\alpha_{01} \equiv\ & x_1 = 0 \wedge x_3 = 1 \wedge x_4 = max \quad \wedge \\ & w_1 = 0 \wedge w_2 = x_2 \wedge w_3 = 1 \wedge w_4 = max\end{aligned}$$

行列对中的 1（$g_i、\ c_i$）：

$$\begin{aligned}\alpha_{02} \equiv\ & x_1 = max \wedge x_3 = 1 \wedge \mathrm{SUC}(x_4, max) \quad \wedge \\ & w_1 = max \wedge w_2 = x_2 \wedge w_3 = 1 \wedge w_4 = max\end{aligned}$$

行列对中的 1（$h_i,\ c_i$）：

$$\begin{aligned}\alpha_{03} \equiv\ & x_1 = max \wedge x_3 = 1 \wedge x_4 = max \quad \wedge \\ & w_1 = max \wedge w_2 = x_2 \wedge w_3 = 1 \wedge w_4 = max\end{aligned}$$

行列对中的 1（y_j，c_i）：

$$\begin{aligned}\alpha_1 \wedge \lambda_1 \equiv \quad & x_1 = 0 \wedge x_3 = 1 \wedge \mathrm{SUC}(x_4, max) \quad \wedge \\ & w_1 = max \wedge w_3 = 1 \wedge w_4 = max \quad \wedge \\ & P(x_2, w_2)\end{aligned}$$

行列对中的 1（z_j，c_i）：

$$\begin{aligned}\alpha_2 \wedge \lambda_2 \equiv \quad & x_1 = 0 \wedge x_3 = 1 \wedge x_4 = max \quad \wedge \\ & w_1 = max \wedge w_3 = 1 \wedge w_4 = max \quad \wedge \\ & N(x_2, w_2)\end{aligned}$$

令

$$\alpha_0 \equiv \alpha_{00} \vee \alpha_{01} \vee \alpha_{02} \vee \alpha_{03},$$

我们得到

$$\varphi_1 \equiv \alpha_0 \vee (\alpha_1 \wedge \lambda_1) \vee (\alpha_2 \wedge \lambda_2).$$

不难看出，这个公式满足投影公式的性质。

$\varphi_2(w_1, w_2, w_3, w_4)$ 定义了 $L^{\rho_1(\mathcal{A})}$。回想一下，目标为 $t = 11\ldots133\ldots3$，因此对于 1，我们只需找到对应的列；对于 3，则需要在每个对应于 c_i-列的 1 左侧写入一个 1。总之，列 $(0, j, 1, n-1)$、$(n-1, i, 1, n-1)$ 和 $(n-1, i, 1, n-2)$ 中必定有 1。

$$\begin{aligned}\varphi_2 \equiv \quad & w_1 = 0 \wedge w_3 = 1 \wedge w_4 = max \quad \vee \\ & w_1 = max \wedge w_3 = 1 \wedge w_4 = max \quad \vee \\ & w_1 = max \wedge w_3 = 1 \wedge \mathrm{SUC}(w_4, max)\end{aligned}$$

因此，φ_2 作为一个数值公式，满足所需性质。

现在，我们解释这个归约为何有效。请注意，我们的归约中的目标 t与原始表 1 中的目标 t 并不完全相同，因为多出了几列 0。不过，我们认为，读者参照原始表会更容易理解证明。

假设存在一个真值赋值，使 3SAT 实例中的每个子句都成立。让我们在相应的子集和实例中构造集合 B'。如果变量x_j被赋值为 1，就将 y_j放入 B'，也就是说，将对应于行 $(0, j, 1, n-2)$ 的元素放入 B'；否则，将 z_j（对应于行 $(0, j, 1, n-1)$）的元素放入 B'。把目前选取的元素相加后，对应于列 $(0, j, 1, n-1)$ 的每个数位上都会是 1。

现在，鉴于每个子句都已满足，带有标签（$n-1$，i，1，$n-1$）的任意列（对应于 c_i）上的当前总和是一个介于 1 和 3 之间的整数；如果不是 3，我们就加上

g_i 或 h_i，以使其变为 3。由于 11 在二进制中表示 3，因此现在我们的总和的列（$n-1$, i, 1, $n-2$）和（$n-1$, i, 1, $n-1$）中都有 1。这个总和正是目标 t。
反之，假设存在 $B' \subseteq B$，使其元素的大小之和等于目标 t。这意味着，对于每个 j，y_j 和 z_j 中恰好有一个属于 B'。如果 $y_j \in B'$，我们将 x_j 赋值为 1；否则，将其赋值为 0。
令 c_i 为 3-CNF公式中的一个子句。注意，其对应的列是带有标签（$n-1$, i, 1, $n-1$）的列。该列中恰好有五个 1。另一方面，目标 t 在列（$n-1$, i, 1, $n-2$）和（$n-1$, i, 1, $n-1$）中含有 1。因此，列（$n-1$, i, 1, $n-1$）中的三个 1 恰好构成该和；其中至多有两个来自大小为 g_i 和 h_i 的元素，所以至少有一个来自 y_j 或 z_j，对于某个 j。如果它来自前者，则将 x_j 赋值为 1，并且它出现在 c_i 中；否则，将 x_j 赋值为 0，并且 $\neg x_j$ 出现在 c_i 中。在这两种情况下，c_i 都得到满足。

总之，我们证明了子集和的一个受限版本可通过 fops 实现 NP难。下一节中，我们将假设每个实例都具有这一限制中的形式。

3.1 划分问题可通过 FOPS 实现 NP难

现在，我们给出子集和 $\leq_{fop}$ 划分问题的证明。划分问题所用的词汇表为 $< T^2 >$；那么，$T(i,j)$ 的解释方式与 $W(i,j)$ 相同。基本上，除了构造大小 $s(b_1)$ 和 $s(b_2)$ 之外，我们还将在归约 ρ_1 输出的子集和实例中的 W 矩阵里添加一些全为 0 的列和行。

我们假设子集和的实例具有其像中 ρ_1所对应的相同形式；特别要注意，k 是与 W相关联的方形二进制矩阵中的一个非零列，当且仅当 $W(k,k)$；这是因为赋给 (z_j, j) 和 (h_i, c_i) 行列对的标签。此外还要注意，ρ_1 像中的每个结构的大小都是 $n \geq 16$；原因是 ρ_1 是四元的，并且每个 3SAT 实例（结构）的大小都大于或等于 2。因此，子集和限制中任何结构的论域都包含 4,3、和 2。

我们的有限元运算 ρ_2 是二元的。我们采用标准归约。对于 SUBSET-SUM 限制中的每个词汇表为 $< W^2, L^1 >$ 的结构 $\mathcal{B}$，我们按情况定义 $T^{\rho_2(\mathcal{B})}$。首先，复制矩阵 W 中的 1。接下来的阶段中，我们定义 $s(b_1)$ 和 $s(b_2)$。

定义 ρ_2 的公式：

$\varphi_0 \equiv$**真**。
$\varphi_1(x_1, x_2, w_1, w_2)$ 定义 $T^{\rho_2(\mathcal{B})}$。
将矩阵 W 中的 1 复制到矩阵 T：

$$x_2 = 0 \wedge w_2 = 4 \wedge W(x_1, w_1)$$

因此，如果矩阵的第 i 行、第 j 列中有一个 1，那么这个 1 就会被复制到矩阵第 $(i, 0)$ 行、第 $(j, 4)$ 列的元素中。

到目前为止，矩阵中已有若干非零列。此外，这些非零列之间至少隔着四列全为 0 的列。事实上，如果 $k < j$ 和 $(j,\ 4)$ 和 $(k,\ 4)$ 是相邻的两列非零列，那么它们之间的列 $(j,\ 3)$，$(j,\ 2)$，$(j,\ 1)$，和 $(j,\ 0)$ 都只含有 0。

现在我们将焦点转向 $s(b_1)$。这个大小将放置在第 (0,1) 行。请注意，这一行与前一阶段的各行不同。回想一下标准归约中的 $s(b_1) = 3333 \ldots 3777 \ldots 7$。我们先处理 3。3 在二进制中是一个 1 后面跟着另一个 1。

如果 T 矩阵的第 $(j,4)$ 列中有一个 1，且它来自前一阶段，那么我们希望在 $(0, 1, j, 4)$ 和 $(0,1, j,3)$ 行-列对中出现 1。下面的公式表达了我们的意图。这里我们利用了这样一个事实：当且仅当该矩阵的 (k, k) 项中有一个 1 时，k 才是 W 矩阵的非零列。另外请注意，3333 ... 3777 ... 7 中的 3 恰好位于$s(b_1)$的十进制表示的左半部分。

$$\begin{aligned} &x_1 = 0 \wedge x_2 = 1 \wedge w_2 = 4 \wedge \beta_{left-half}(w_1) \wedge W(w_1, w_1) \quad \vee \\ &x_1 = 0 \wedge x_2 = 1 \wedge w_2 = 3 \wedge \beta_{left-half}(w_1) \wedge W(w_1, w_1) \end{aligned}$$

我们之后会定义 $\beta_{left-half}(w_1)$。

现在，我们来处理 $s(b_1)$ 中的 7。$= 3333 \ldots 3777 \ldots 7.$ 7 在二进制中是 111，因此我们现在要在连续的三列中放入三个 1。以下公式实现了这一点，理由与上面给出的理由类似。

$$\begin{aligned} &x_1 = 0 \wedge x_2 = 1 \wedge w_2 = 4 \wedge \beta_{right-half}(w_1) \wedge W(w_1, w_1) \quad \vee \\ &x_1 = 0 \wedge x_2 = 1 \wedge w_2 = 3 \wedge \beta_{right-half}(w_1) \wedge W(w_1, w_1) \quad \vee \\ &x_1 = 0 \wedge x_2 = 1 \wedge w_2 = 2 \wedge \beta_{right-half}(w_1) \wedge W(w_1, w_1) \end{aligned}$$

采用类似的策略，我们现在定义 $s(b_2)$。对应的行是带有标签 $(1,\ 1)$。 在原始归约中，$s(b_2) = 3333 \ldots 3888 \ldots 8.$ 下面说明如何定义 3。

$$\begin{aligned} &x_1 = 1 \wedge x_2 = 1 \wedge w_2 = 4 \wedge \beta_{left-half}(w_1) \wedge W(w_1, w_1) \quad \vee \\ &x_1 = 1 \wedge x_2 = 1 \wedge w_2 = 3 \wedge \beta_{left-half}(w_1) \wedge W(w_1, w_1) \end{aligned}$$

现在，我们来定义 8（记住，8 的二进制表示是 1000）。

$$x_1 = 1 \wedge x_2 = 1 \wedge w_2 = 1 \wedge \beta_{right-half}(w_1) \wedge W(w_1, w_1)$$

请注意，上面的公式 $w_2 = 3$ 当然是以下内容的缩写

$$(\exists t_1)\left(\mathrm{SUC}\left(1, t_1\right) \wedge \mathrm{SUC}\left(t_1, w_2\right)\right)$$

或等价的数值一阶公式。相关公式也同样如此。

现在，当且仅当 $k \leq {}^{n-2}_{2}$ 或 $k < {}^{n-1}_{2}$ 时，$\beta_{left-half}(k)$ 成立，其中这些分数都是整数。下面给出该公式的定义。

$$\beta_{left-half}(w_1) \equiv (\exists t_1)\,(\exists t_2)\,(\mathrm{SUC}\,(t_1, max) \wedge \mathrm{PLUS}\,(t_2, t_2, t_1) \wedge w_1 \leq t_2) \quad \vee \\ (\exists t_2)\,(\mathrm{PLUS}\,(t_2, t_2, max) \wedge w_1 < t_2)$$

当且仅当 $k > {}^{n-2}_{2}$ 或 $k \geq {}^{n-1}_{2}$ 时，$\beta_{right-half}(k)$ 成立。下面给出定义。

$$\beta_{right-half}(w_1) \equiv (\exists t_1)\,(\exists t_2)\,(\mathrm{SUC}\,(t_1, max) \wedge \mathrm{PLUS}\,(t_2, t_2, t_1) \wedge w_1 > t_2) \quad \vee \\ (\exists t_2)\,(\mathrm{PLUS}\,(t_2, t_2, max) \wedge w_1 \geq t_2)$$

注意，$\beta_{left-half}(w_1)$ 和 $\beta_{right-half}(w_1)$ 是互斥的公式。

总之，定义该归约的公式可以表示为

$$\varphi_1 \equiv \alpha_0 \vee \alpha_1 \wedge \lambda_1 \vee \alpha_2 \wedge \lambda_2,$$

其中 $\alpha_0 \equiv$ **假**，λ_1 是 $W(x_1,\ w_1)$，而 λ_2 是 $W(w_1,\ w_1)$。α_2 是用于定义 $s(b_1)$ 和 $s(b_2)$ 的数值子公式的析取。不难看出，α_1 和 α_2 互斥。

我们将通过计算非零列中各项的和来证明该归约有效。

在继续之前，定义 $F\colon E \subseteq B \to D \subseteq A$ 为一个函数，该函数将 SUBSET-SUM 实例中大小非零的元素映射到划分问题实例中的对应元素。后者是归约第一阶段生成的元素。

现在，假设 SUBSET-SUM 实例中存在 $B' \subseteq B$，使得 B' 中元素大小之和恰好为 t。我们希望证明，通过我们的归约输出的实例是划分问题的一个是实例。定义 $(A', A \setminus A')$ 为 A （在划分问题实例中）的划分，其中

$$A' = F\left(E \cap B'\right) \cup \{b_1\}.$$

因此，A' 中元素的大小之和为 $t + s(b_1)$：

$$\begin{array}{cccccccccccccccccccccccccccc}
 & & & 1 & \ldots & & & 1 & \ldots & & & 1 & \ldots & & & 1 & 1 & \ldots & & & 1 & 1 & \ldots & & & 1 & 1 \\
+ & & & & & & & & & & & & & & & & & & & & & & & & & \\
 & & 1 & 1 & \ldots & & 1 & 1 & \ldots & & 1 & 1 & \ldots & & 1 & 1 & 1 & \ldots & & 1 & 1 & 1 & \ldots & & 1 & 1 & 1 \\
\hline
 & 1 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & 0 & 1 & 0 & \ldots & 1 & 0 & 1 & 0 & \ldots & 1 & 0 & 1 & 0.
\end{array}$$

注意，划分问题实例中大小的总和为

$$\sum_{a \in F(E)} s(a) \quad + s(b_1) \quad + s(b_2).$$

在原始归约中，左侧的和等于 2222...2555...5，它是将子集和实例中的所有大小相加得到的。因此，上述总和按相应顺序为：

$$
\begin{array}{cccccccccccccccccccccccccccccccccccc}
 & 1 & 0 & \dots & & 1 & 0 & \dots & & 1 & 0 & \dots & & 1 & 0 & 1 & \dots & & 1 & 0 & 1 & \dots & & & 1 & 0 & 1 \\
+ & & & & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & 1 & \dots & & 1 & 1 & 1 & \dots & & & 1 & 1 & 1 \\
+ & & & & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 1 & 0 & 0 & \dots & 1 & 0 & 1 & 0 & 0 & \dots & 1 & 0 & 1 & 0 & 0.
\end{array}
$$

如果我们从最后一个总和中减去 A' 中元素大小之和，就得到 $A \setminus A'$ 中元素大小之和：

1000...1000...1000...10100...10100...10100100...100...100...1010...1010...1010100...100...100...1010...1010...1010,因此我们得出结论：归约输出的划分问题实例确实是一个是实例。

反过来，假设存在一个 A 的划分 $(A', A \setminus A')$，使得对应大小的和相同。这意味着

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a),$$

根据上文讨论，这些和都必须是

1 0 0 ... 1 0 0 ... 1 0 0 ... 1 0 1 0 ... 1 0 1 0 ... 1 0 1 0。如果 $s(b_1)$ 和 $s(b_2)$ 都属于 A'，那么

$$\begin{aligned}
\sum_{a \in A'} s(a) &\geq s(b_1) + s(b_2) \\
&> \sum_{a \in A \setminus \{b_1, b_2\}} s(a) \\
&\geq \sum_{a \in A \setminus A'} s(a),
\end{aligned}$$

因为 $s(b_1) + s(b_2)$ 可表示为

$$
\begin{array}{cccccccccccccccccccccccccccccccccccc}
 & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & 1 & \dots & & 1 & 1 & 1 & \dots & & & 1 & 1 & 1 \\
+ & & & & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 1 & \dots & & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 \\
\hline
1 & 1 & 0 & \dots & 1 & 1 & 0 & \dots & 1 & 1 & 0 & \dots & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1,
\end{array}
$$

while $\sum_{a \in A \setminus \{b_1, b_2\}} s(a)$ is

1000...1000...1000...10100...10100...10100-110...110...110...1111...1111...11
1110...10...10..101..101...101.

因此，如果 $s(b_1)$ 和 $s(b_2)$ 都属于 A'，就会产生矛盾。由此可知，$s(b_1)$ 和 $s(b_2)$ 必须属于划分问题的不同部分。不失一般性，假设 $b_1 \in A'$ 和 $b_2 \in A \setminus A'$。

注意，$\sum_{a \in A'} s(a) - s(b_1)$ 是

$$\begin{array}{cccccccccccccccccccccccccccc}
1 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & 0 & 1 & 0 & \ldots & 1 & 0 & 1 & 0 & \ldots & 1 & 0 & 1 & 0 \\
- & & & & & & & & & & & & & & & & & & & & & & & & & \\
 & 1 & 1 & \ldots & & 1 & 1 & \ldots & & 1 & 1 & \ldots & & 1 & 1 & 1 & \ldots & & 1 & 1 & 1 & \ldots & & 1 & 1 & 1 \\
\hline
 & & 1 & \ldots & & & 1 & \ldots & & & 1 & \ldots & & & 1 & 1 & \ldots & & & 1 & 1 & \ldots & & & 1 & 1.
\end{array}$$

现在，定义集合

$$B' = F^{-1}\left(D \cap \left(A' \setminus \{s(b_1)\}\right)\right).$$

这是输入子集和实例中大小非零的元素集合，这些元素通过 F 映射到 $A' \setminus \{s(b_1)\}$ 中大小非零的元素。因此，B' 中元素的大小之和为

$$\begin{array}{ccccccccccccc}
1 & \ldots & 1 & \ldots & 1 & \ldots & 1 & 1 & \ldots & 1 & 1 & \ldots & 1 & 1, \\
\end{array}$$

因为每一列中各项之和保持不变。总之，$\sum_{a \in B'} s(a) = t$，因此该子集和实例是一个肯定实例。由此完成了证明，说明子集和 $\leq_{fop}$ 划分问题。

参考文献

[ABI97] Eric Allender、Jos´e Balc´azar 和 Neil Immerman。《一阶同构定理》。载于：《*SIAM*计算杂志》26.2（1997年），第 557–567 页。DOI：10 . 1137 / S0097539794270236。电子版：https: / / doi . org / 10 . 1137 / S0097539794270236。URL：https://doi.org/10.1137/S0097539794270236。[CSV84] Ashok K. Chandra、Larry Stockmeyer 和 Uzi Vishkin。《常深度可归约性》。载于：《*SIAM*计算杂志》13.2（1984年），第 423–439 页。DOI：10.1137/0213028。电子版：https://doi.org/10.1137/0213028。URL：https://doi.org/10.1137/0213028。[GJ79] Michael R. Garey 和 David S. Johnson。《计算机与难解性：NP完全性理论指南》。美国：W. H. Freeman & Co.，1979年。ISBN：0716710447。[Imm99] Neil Immerman。《描述复杂性》。施普林格，1999年。DOI：10.1007/978- 1-4612-0539-5。

[MW17] Cody D. Murray 和 R. Ryan Williams。《计算电路复杂性的（非）NP困难性》。载于：《计算理论》13.4（2017年），第 1–22 页。DOI：10 . 4086 / toc . 2017. v013a004。URL：https : / / theoryofcomputing . org/articles/v013a004。[Pa ́u] Risco Pa ́ul。《通过一阶归约，划分问题是否为 NP 完全？》计算机科学 *Stack Exchange*。版本：2025年1月12日。URL：https://cs.stackexchange.com/q/170906。[Sip13] Michael Sipser。《计算理论导论》。圣智学习，2013年。[Ste94] Iain A. Stewart。《通过投影变换得到 NP 完全性》。载于：《数学系统理论》27.2（1994年3月），第 125–157 页。ISSN：0025-5661。DOI：10. 1007/BF01195200。URL：https://doi.org/10.1007/BF01195200。