arXiv:2603.21836v1 [cs.CL] 23 Mar 2026

符号回归的指令集与语言

预印本

Ezequiel López-Rubio*计算机语言与计算机科学系
马拉加大学 Louis Pasteur 大道 35 号 29071 马拉加，
西班牙ezeqlr@lcc.uma.es

Mario Pascual-González计算机语言与计算机科学系 马拉加大学 Louis Pasteur大道35号 29071 马拉加，
西班牙mpascual@uma.es

3月24日 , 2026

摘要

符号回归（SR）中一个根本但在很大程度上尚未得到解决的障碍是结构冗余：每个含有 k 个内部节点的表达式有向无环图，都有 $\Theta(k!)$ 种不同的节点编号方案，它们编码的都是同一个表达式，却各自占据搜索空间中的一个独立位置，并消耗适应度评估，却没有增加多样性。

我们提出了 ISALSR（符号回归的指令集与语言），这是一种表示框架，可将表达式有向无环图编码为由紧凑的双层字母表构成的字符串，并计算剪枝规范字符串——一种完备的带标签有向无环图同构不变量——将所有 $\Theta(k!)$ 种等价表示合并为单一规范形式。
我们在 Nguyen 和 AIFeynman 基准测试套件上，通过实证验证了该表示的五项基本性质：往返保真度、有向无环图无环性、规范不变性与幂等性、求值保持，以及搜索空间缩减。

关键词 符号回归、规范字符串表示、带标签有向无环图、同构不变量、搜索空间缩减

1 引言

符号回归（SR）旨在发现一个闭式数学表达式 $f(x_1, \ldots\ldots, x_m)$ 使其拟合观测数据集 (X, y)，而不预先限制函数形式 *a priori*。与参数回归不同，SR 同时搜索数学结构及其数值参数的空间，这使其在组合上颇具挑战，但也具有独特的可解释性：成功的模型不是黑箱，而是可读公式，可以对其进行检查、简化，并外推至训练范围之外 [Koza, 1992]。这种可解释性推动了其在科学发现 [Udrescu 和 Tegmark、 2020], 工程 [Randall 等人, 2022], 以及机器学习 [Cava 等人, 2021]等领域的应用。

尽管数十年来不断取得进展，符号回归（SR）的搜索空间仍存在结构冗余，而现有方法都未能直接解决这一问题。数学表达式可以表示为带标签的有向无环图（DAG），其中叶节点是输入变量或可学习常数，内部节点则是算术运算或超越运算。对于一个 DAG

*通讯作者。ITIS So ftware。马拉加大学。C/ Arquitecto Franci sco Peñalosa 18，29010，马拉加，西班牙

含有 k 个内部节点的 DAG，存在 $\Theta(k!)$ 种不同的节点编号方案，它们编码的都是同一个表达式。每一种等价表示都会占据搜索空间中的一个独立位置，消耗适应度评估次数和种群名额，却不会增加多样性。对于含有 $k = 8$ 个内部节点的 DAG——这是一个规模适中的表达式——每个结构上独特的候选项都会产生超过 40，000 个冗余副本。目前还没有符号回归方法能在这一粒度上进行去重。

贡献。 我们提出 ISALSR（符号回归的指令集与语言），这是一种表示框架，可将表达式有向无环图编码为指令字符串，并计算出剪枝规范字符串——一种完整的带标签有向无环图不变量——将所有 $\Theta(k!)$ 个等价表示归并为单一规范形式。任何现有的符号回归算法都可以通过在适应度评估前插入一个规范化步骤来扩展为 ISALSR，从而将有效搜索空间缩小 $O(k!)$，且不改变算法的核心逻辑。
ISALSR 与 IsalGraph 有一些相似之处 [Lopez-Rubio, 2025], 这是一个先前提出的表示框架，用于将无标签无向图编码为指令字符串。然而，ISALSR 有着根本性的不同，因为它专用于符号回归所要求的带标签、有向、无环场景。ISALSR 的主要贡献包括：(i) 采用双层指令字母表，同时编码图拓扑结构和运算标签，使字母表上的所有字符串都表示有效有向无环图；(ii) 采用交换编码，以一元分解运算符替代二元非交换运算符（减法、除法），从而简化同构定义；(iii) 提出一种标签感知剪枝回溯算法，高效计算规范字符串；以及 (iv) 证明在所采用的语义下，规范字符串是带标签有向无环图的完整同构不变量，其中包括对唯一剩余的非交换算子（POW）保持操作数顺序。

1.1 相关工作

遗传编程。 自 Koza [Koza以来, 1992]遗传编程（遗传编程）一直是符号回归的主流范式，它通过交叉和变异来演化表达式树种群。遗传编程自然会产生冗余种群：两个表达式相同但树形结构不同的个体会被视为不同个体。语义交叉算子 [Uy 等人, 2011] 通过考虑行为等价性部分解决了这一问题，但无法处理结构同构。规范化与所有遗传编程变体彼此正交，且可以结合使用。

深度学习与强化学习方法。 Petersen等人 [Petersen等人, 2021] 提出深度符号回归（深度符号回归），通过风险偏好策略梯度训练循环神经网络，按序生成表达式树。循环结构同样缺乏检测两个生成序列是否对应同一表达式的机制。Liu 等人 [Liu 等人, 2025]提出 GraphDSR，以有向无环图而非树来表示表达式，从而实现共享子表达式复用，并在 Nguyen 和 AI Feynman基准上取得最先进性能。GraphDSR 的有向无环图表示使冗余问题更加突出：变元数加法和乘法节点的输入存在多种等价排序，而基于有向无环图的方法在不进行去重的情况下可能会探索所有这些排序。ISALSR 的规范字符串直接消除了 GraphDSR 式表达式有向无环图中的这种歧义。

基于图的符号回归。 Xiang 等人 [Xiang 等人指出，在 GraphSR 中，非交换二元运算（减法、除法）会造成操作数顺序歧义，使同构定义变得复杂。他们提出了一种一元分解：$x - y = \text{Add}(x, \text{Neg}(y))$ 和 $x/y = \text{Mul}(x, \text{Inv}(y))$，用可交换二元节点和新的一元运算符替换二元非交换节点。ISALSR 采用了这种分解，使非交换二元运算仅剩 POW。

均匀深度优先搜索（UDFS）。 Kahlmeyer 等人 [Kahlmeyer 等人 2024] 提出了一种基于对表达式有向无环图进行均匀深度优先搜索的符号回归结构化枚举策略，并通过拓扑编号系统地生成候选项。UDFS 表明，对表达式空间进行结构化遍历比随机搜索能覆盖更多内容，但它不会识别或合并结构同构有向无环图。ISALSR 的规范字符串可作为天然的去重键，整合到 UDFS 式枚举中：规范字符串相同的候选项彼此同构，可以跳过。

物理信息与基于语法的符号回归。 Udrescu 和 Tegmark [Udrescu 和 Tegmark, 2020]利用物理对称群和量纲分析来约束 AI Feynman基准的搜索空间。基于语法的遗传编程方法通过上下文无关文法施加句法约束，以剪除无效表达式。这些方法通过限制生成的表达式来缩小搜索空间，而 ISALSR 无需剪枝便不存在无效表达式，因为定义的指令字母表中的每个字符串都对应一个有效有向无环图。此外，ISALSR 还通过识别生成的字符串中哪些彼此等价，进一步缩小搜索空间。

BINGO。 Randall 等人。 [Randall 等人， 2022] 推出了 BINGO（通过优化提升隐式非线性基因表达编程），这是一款由美国国家航空航天局开发、面向工程和科学应用的开源符号回归工具。BINGO 将年龄-适应度帕累托优化与自动微分以及基于 BFGS 的常数优化相结合，在航空航天应用中取得了有竞争力的表现。与所有基于遗传编程的方法一样，BINGO 的基于种群的搜索容易受到 $O(k!)$ 结构冗余的影响，而 ISALSR 消除了这种冗余；其高效的常数优化器和模块化架构使其自然适合承载 ISALSR 规范化。

SRBench。 La Cava 等人 [Cava 等人， 2021]提供了一个大规模基准测试，在 PMLB 收录的 252 个真实世界回归数据集上对当代符号回归方法进行比较，涵盖遗传编程、深度符号回归和线性基线方法等。他们的全面评估表明，没有任何一种方法能在所有数据集上占据优势，这促使研究者寻求相互正交的改进，例如 ISALSR 所实现的搜索空间缩减。

1.2 论文结构

本文其余部分结构如下。第 2 节定义了 IsalSR 指令集、字符串到有向无环图和有向无环图到字符串算法、剪枝规范字符串以及同构不变性。第3节介绍实验设计，包括 Nguyen、AI Feynman 和 SRBench 基准测试套件。第4节展示对规范表示五项基本性质的实证验证，并给出关于同构有向无环图均表示为单一指令字符串的实证证明（第 4.2节）。第5节总结本文贡献并展望未来研究方向。

2 方法论

本节介绍 IsalSR 的形式化基础。我们定义指令集（§2.1）、字符串到有向无环图解码器（§2.2）、有向无环图到字符串编码器（§2.3）、剪枝规范字符串（§2.4），并提出猜想，认为它是带标签有向无环图同构下的完备图不变量（§2.5）。

2.1 IsalSR 指令集

ISALSR 将符号表达式表示为带标签有向无环图（有向无环图），并将其编码为紧凑双层字母表上的字符串。在定义字母表之前，我们先确定底层数据结构。

定义 2.1 （带标签有向无环图）。带标签有向无环图是一个元组 $D=(V,E,\ell,\delta)$，其中：

(i) $V=\{0,1,\ldots,n-1\}$ 是一个有限节点集，其全序由整数索引确定；(ii) $E\subseteq V\times V$ 是一个有向边集，使得 (V,E) 是无环的；(iii) ℓ：$V\to\mathcal{T}$ 为每个节点指定类型集 $\mathcal{T}$（表 1）中的一种操作类型；(iv) $\delta: V\to\mathbb{Z}\cup\mathbb{R}$ 存储可选的逐节点元数据：VAR 节点的变量索引 i（$\delta(v)=i$）、CONST 节点的常数值，以及所有其他节点的零值。

边 $(u,\ v)\in E$ 表示“u 为 v提供输入”（数据流方向）。 v的入邻居 为 $N^{-}(v)=\{u:\ (u,\ v)\in E\}$；出邻居 为 $N^{+}(v)=\{u:\ (v,\ u)\in E\}$。

POW 的操作数顺序。 每个节点 v 都维护一个有序输入列表 $\sigma(v)=(s_1,\ s_2,\ \ldots)$，按照连接边添加到有向无环图中的顺序记录 v 的入邻居。POW 是唯一的二元非交换运算：若$\sigma(v)=(u_1,\ u_2)$，则其求值约定为

$$\mathrm{eval}(v)=\mathrm{eval}(u_1)^{\,\mathrm{eval}(u_2)},\tag{1}$$

其中 u_1 是底数， u_2 是指数。添加到 v 的第一条边决定 u_1：Vℓ/vℓ 创建令牌会先添加创建边，因此创建边的源始终是 $\sigma(v)[0]$, 底数。后续的 C/c 指令会附加指数 $\sigma(v)[1]$。所有其他运算节点（ADD、MUL 和所有一元运算）都是可交换的，因此其入邻居的顺序不会影响求值。

机器状态。ISALSR 机器状态是四元组 $\mathcal{S}=(D,\ L,\ p,\ q)$，其中 D是部分构建的带标签有向无环图， L 是循环双向链表 (CDLL)，其元素是来自 V的节点标识符； p 是主指针在 CDLL 中的索引， q 是次指针。在 CDLL 位置 r 处的图节点可通过 val(r) $\in V$获取。反过来，图节点 v 在 CDLL 中的位置可通过 ptr(v) $\in V$获取。

定义 2.2 (IsalSR 指令集 Σ_{SR})。 设 $\mathcal{L}=\{+,$ *、 -, /、 s、 c、 e、 l、 r、 ^、 a、 k$\}$ 为标签字符集。字母表 Σ_{SR} 由四类令牌组成：

表 1：IsalSR 操作类型、标签字符、元数和求值语义。标记为 † 的操作使用数值保护实现。Pow 是唯一的二元非交换运算：其操作数顺序由 $\sigma(v)$（边插入顺序）给出，其中 $\sigma(v)[0]$ 为底数， $\sigma(v)[1]$ 为指数。Neg 和 Inv 是对减法和除法进行可交换编码的一元分解运算符（见 §2.1）。

标签 ℓ	Type	元数	求值
+	Add	多元（≥2）	$\sum_{u\in N^-(v)}$ 求值（u） （可交换）
*	Mul	多元（≥2）	$\prod_{u\in N^-(v)}$ 求值(u) (可交换)
g	Neg	一元 (1)	−求值(u)
i	Inv	一元 (1)	1 / 求值(u) †
s	Sin	一元 (1)	正弦(eval(u))
c	Cos	一元 (1)	余弦(eval(u))
e	Exp	一元 (1)	exp(eval(u)) †
l	Log	一元 (1)	ln(求值(u)) †
r	Sqrt	一元 (1)	$\sqrt{\text{求值}(u)}$ †
a	Abs	一元 (1)	|求值(u)|
^	Pow	二进制（2）	求值($\sigma(v)[0]$)$^{\text{eval}(\sigma(v)[1])}$ †
k	const	叶节点（0）	可学习实常数
（固定）	Var	叶节点 (0)	输入变量 x_i （预先插入）

1. **主指针移动**（单个字符）：N（前进到循环双向链表的下一个元素），P（后退到循环双向链表的上一个元素）。

2. **次指针移动**（单个字符）：n（前进），p（后退）。
3. **边创建**（单个字符）：C 创建有向边 val(p) → val(q)；c 创建 val(q) → val(p)。如果该边会形成环，则会静默跳过。

4. **空操作**（单个字符）：W。
5. **带标签节点插入**（双字符标记）：Vℓ 创建一个类型为 ℓ的新节点 v_{new}，添加有向边 val(p) → v_{new}，并将 v_{new} 插入循环双向链表中紧接主指针位置之后（指针不会移动）。vℓ 使用次指针执行相同操作 q。

词汇表包含 7 个单字符令牌和 24 个复合令牌（$2\times|\mathcal{L}|$），共计 **31** 个令牌。

减法和除法的交换编码。 两个一元运算符 Neg (标签 g，$-x$)和 Inv (标签 i,$1/x$)通过受 GraphSR 启发的交换分解来实现减法和除法 [Xiang 等人,2025]:

$$x-y \;=\; \textsc{Add}\big(x,\ \textsc{Neg}(y)\big), \qquad x/y \;=\; \textsc{Mul}\big(x,\ \textsc{Inv}(y)\big).$$

ADD 和乘法都具有可交换性，因此这些节点的输入可以互换，无需跟踪操作数顺序。受保护的倒数为 INV(x) $=1/x$，当 $|x|>\varepsilon$ 时如此，否则为 1。

初始状态。 给定 $m\geq 1$个输入变量，机器初始化如下：

- $V_0=\{x_1,\ldots,x_m\}$ 节点 ID 按变量索引顺序分配为 $0,\ldots,m-1$； $\ell(x_i)$ =变量， $\delta(x_i)=i$。
- $E_0=\emptyset$。
- $L_0=[x_1,\ldots,x_m]$（循环的，按变量索引顺序排列）。
- $p_0=q_0=$ ptr(x_1)。

由于节点 ID 按插入顺序分配——从 0 开始， x_1——因此，之后插入的任何节点都会获得严格更高的 ID。这一约定使 S2D 生成的每个有向无环图中的方程（1）的操作数顺序语义都有明确定义且具有确定性。

无环性不变量。每处理完一个 token，(V,E) 都保持无环。对于 Vℓ/vℓ，新节点在创建时没有出边，因此向其添加边不会形成环。对于 C/c，插入前会执行 BFS 可达性测试；如果插入会导致形成环，则会静默跳过该指令。

2.2 字符串到有向无环图算法（S2D）

字符串到有向无环图（*S2D*）算法是一种确定性状态机，可将 IsalSR 字符串 $w\in\Sigma_{\text{SR}}^*$ 解码为带标签有向无环图。LabeledDAG 数据结构会为每个节点 v追踪一个有序输入列表 $\sigma(v)$，该列表记录

按边插入顺序排列的入邻居。对于唯一的二元非交换运算 Pow，此列表通过方程（1）编码求值语义：$\mathrm{v}\ell$/$\mathrm{v}\ell$ 创建令牌首先添加创建边，从而确定 $\sigma(v)[0]$（底数）；随后 C/c令牌设置 $\sigma(v)[1]$（指数）。

定义 2.3 (S2D 执行)。给定 $w \in \Sigma_{\mathrm{SR}}^*$ 和 $m \geq 1$，令 $(t_1, \ldots, t_k) =$ 词元化(w) 由双层扫描器（定义 2.2）生成的令牌序列。从 $\mathcal{S}_0$开始，令牌 t_j 根据表 $\mathcal{S}_{j-1} \to \mathcal{S}_j$ 中的规则产生状态转移 2。输出为 $\mathrm{S2D}(w, m) = D_k$。

表 2：伪代码 f or 字符串到有向无环图（S2D）算法。每个节点 v 维护一个有序输入列表 $\sigma(v)$；边 s 会被追加到其中，按 i 插入顺序，为非交换二元节点确立操作数语义。

算法 S2D$(w,\ m)$

输入： 指令字符串 w；变量数量 $m \geq 1$
输出： 带标签有向无环图 $D = (V, E, \ell, \delta, \sigma)$

// 初始化
创建 m 个 VAR 节点 $x_1, \ldots, x_m$，其标识符为 $0, \ldots, m-1$；设置 $E \leftarrow \emptyset$；对所有 v，令 $\sigma(v) \leftarrow []$
插入 $x_1, \ldots, x_m$ 按变量索引顺序插入到循环双向链表 L 中
$p \leftarrow \mathrm{ptr}(x_1)$; $q \leftarrow \mathrm{ptr}(x_1)$
对于每个词元 $t \in$ 词元化(w)**执行**
  如果 $t =$ 自然数 **则** $p \leftarrow$ 下一个(L, p)
  elif $t = \mathtt{P}$ **则** $p \leftarrow \mathtt{prev}(L, p)$
  elif $t =$ 自然数 **then** $q \leftarrow \mathtt{next}(L, q)$
  elif $t = \mathtt{p}$ **则** $q \leftarrow \mathtt{prev}(L, q)$
  elif $t = \mathtt{C}$**则**
    $u \leftarrow$ 值(L, p); $v \leftarrow$ 值(L, q)
    if v 无法从 u 到达 D **then**
      $E \leftarrow E \cup \{(u, v)\}$; $\sigma(v)$ += $[u]$ // 将 u追加到 v的有序输入列表
  否则如果 $t =$ **则**
    $u \leftarrow \mathrm{val}(L,\ q)$; $v \leftarrow \mathrm{val}(L,\ p)$
    if v 无法从 u到达，则在 D **then**
      $E \leftarrow E \cup \{(u, v)\}$; $\sigma(v)$ += $[u]$
  否则如果 $t = \mathtt{w}$**则(**空操作**)**
  否则如果 $t = \mathrm{v}\ell$**则**
    $u \leftarrow \mathrm{val}(L,\ p)$
    创建类型为 ℓ 且具有下一个可用标识符的新节点 v；$V \leftarrow V \cup \{v\}$; $\sigma(v) \leftarrow []$
    $E \leftarrow E \cup \{(u,\ v)\}$; $\sigma(v)$ += $[u]$ // u变为 $\sigma(v)[0]$, 第一个操作数
    在 L, p, v之后插入_; p保持不变
  否则，如果 $t = \mathrm{v}\ell$，**则**
    $u \leftarrow \mathrm{val}(L,\ q)$
    创建一个类型为 ℓ、标识符为下一个可用值的新节点 v; $V \leftarrow V \cup \{v\}$; $\sigma(v) \leftarrow []$
    $E \leftarrow E \cup \{(u, v)\}$; $\sigma(v)$ += $[u]$
    在 L、q、v 之后插入 _; q保持不变
返回 $D = (V,\ E,\ \ell,\ \delta,\ \sigma)$

2.3 有向无环图到字符串算法（D2S）

有向无环图到字符串（D2S）算法是一种贪心编码器。给定一个带标签有向无环图 D，它会生成一个字符串 w，使得 $\mathrm{S2D}(w,\ m) \cong D$，其中 $\cong$ 表示带标签有向无环图同构（见定义 2.9）。D2S 会维护一个并行的输出有向无环图 D'，以及双向索引映射 i2o：$V_D \to V_{D'}$ 和 o2i：$V_{D'} \to V_D$；初始化时，按变量索引顺序映射各个 m VAR 节点。一个关键的结构属性是，D' 中的节点 ID 按插入顺序分配；因此，D'中任意两个非变量节点的相对顺序——以及非交换二元运算的操作数语义——完全由 D2S 插入这些节点的顺序决定。

定义 2.4(螺旋位移集合)。对于当前规模为 n 的输出有向无环图，候选位移集合为

$$\mathcal{P}_n = \{(a, b) \in \mathbb{Z}^2 : -n \leq a, b \leq n\},$$

按键 $(|a| + |b|,\ |a|, (a, b))$ 排序（总成本优先，其次按字典序打破平局）。这种螺旋枚举确保优先尝试成本较低的位移。

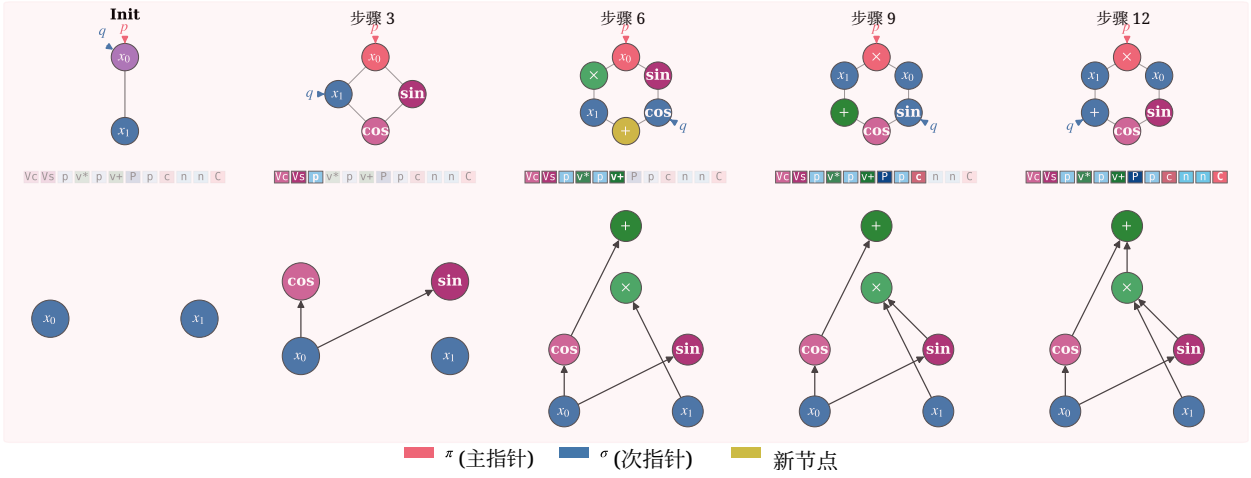


图1：规范字符串 VcVspv*pv+PpcnnC 的字符串到有向无环图的执行过程。有向无环图逐步扩展，每条指令都会插入带标签的节点（V/v）或有向边（C/c），主指针（π）和次指针（σ）则用于遍历循环双向链表。

贪心循环遍历 $\mathcal{P}_n$，并针对每一对 (a, b)，按严格的优先级顺序求值四种候选操作：Vℓ（通过主指针插入）、vℓ（通过次指针插入）、C、c。输出第一个有效操作，更新指针，然后继续循环，直至放置完 D 的所有节点和边。

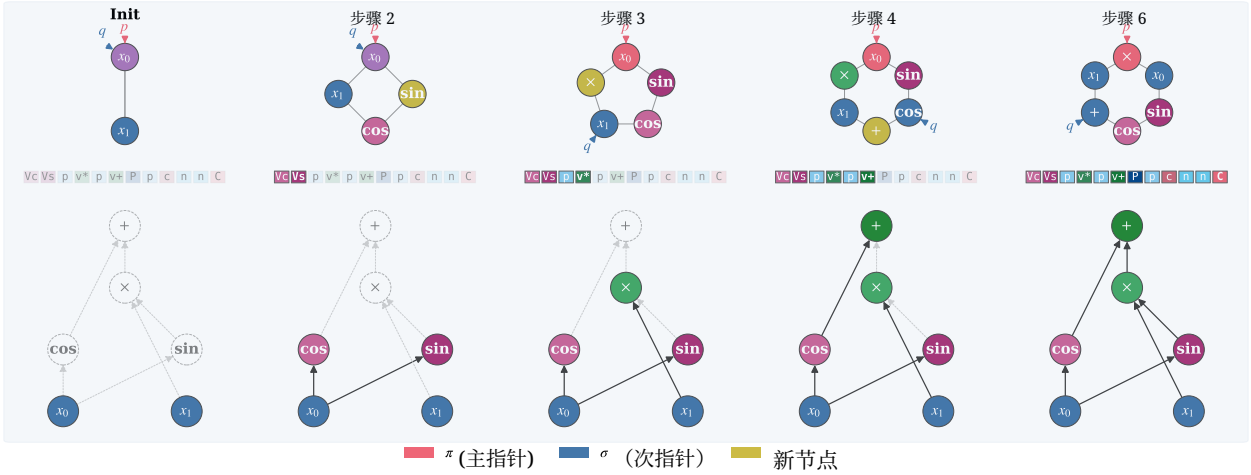


图2：通过从 x_0 开始贪心遍历，对 $\sin(x_0) \cdot x_1 + \cos(x_0)$ 进行有向无环图到字符串的编码。幽灵节点（虚线）表示尚未编码的部分；随着所有节点和边均被遍历，输出的令牌序列逐渐收敛为规范字符串。

2.4 剪枝规范字符串

贪心 D2S 字符串取决于输入有向无环图的内部节点编号；两个结构等价但编号不同的有向无环图可能产生不同的字符串，而且——至关重要的是——不同的编号可能会颠倒非交换二元运算的操作数顺序。规范字符串通过在每个 Vℓ/vℓ 分支点探索所有有效的插入顺序，消除了这种依赖，从而也探索了非交换节点所有可能的操作数顺序。

定义2.5（有效字符串集合）。令 $\mathcal{W}(D)$ 表示在 D 上执行 D2S过程时可生成的所有字符串的集合，其中在每个具有 $k \geq 2$ 个尚未插入的暂定指针位置出邻居的 Vℓ/vℓ 分支点处，都可以选择其中任意一个邻居作为下一个插入的节点。

表 3：有向无环图到字符串（D2S）算法的伪代码。moves(a)：输出 $|a|$ 个 N（若 $a>0$）或 P（若 $a<0$）。moves$_s$(b)：对次指针，类似地输出 n/p。

算法 D2S$(D,\, m)$

输入： 带标签有向无环图 $D=(V,E,\ell,\delta)$；变量数量 m
输出： IsalSR 指令字符串 w
$w \leftarrow \varepsilon$；创建空的输出有向无环图 D'；i2o $\leftarrow \{\}$；o2i $\leftarrow \{\}$
按变量索引顺序将所有 m VAR 节点从 D 映射到 D'；插入循环双向链表 L
$p \leftarrow$指针（x_1）；　$q \leftarrow$ 指针（x_1）
$N_{\text{left}} \leftarrow |V| - m$；　$E_{\text{left}} \leftarrow |E|$
当 $N_{\text{left}} > 0$**或** $E_{\text{left}} > 0$ **时执行**
　对于每个 $(a,\ b) \in \mathcal{P}_{|V'|}$ **执行**　　// 按成本排序
　　$\tilde{p} \leftarrow$遍历($L,\ p,\ a$)；　$u_p \leftarrow$ o2i[值($L, \tilde{p}$)]
　　如果 $N_{\text{left}} > 0$ **且** ∃ 未插入 $v \in N^+(u_p)$ **</style}则**
　　w += 移动(a)Vℓ_v；将 v 插入 D' 和循环双向链表中的 $\tilde{p}$之后；　$p \leftarrow \tilde{p}$
　　更新 i2o，o2i；　$N_{\text{left}}--$；　$E_{\text{left}}--$；**中断**
　　$\tilde{q} \leftarrow$遍历($L,\ q,\ b$)；　$u_q \leftarrow$ o2i[值($L, \tilde{q}$)]
　　如果 $N_{\text{left}} > 0$ **且** ∃ 未插入 $v \in N^+(u_q)$ **则**
　　w += 移动$_s$(b)Vℓ_v；将 v 插入 D' 和 CDLL 之后 $\tilde{q}$；　$q \leftarrow \tilde{q}$
　　更新 i2o，o2i；　$N_{\text{left}}--$；　$E_{\text{left}}--$；**中断**
　　如果$(u_p,\ u_q) \in E$ **且**(i2o$[u_p]$, i2o$[u_q]$) $\notin E'$ **则**
　　w += 移动(a) 移动$_s$(b) C；　$E' \leftarrow E' \cup \{$(i2o$[u_p]$, i2o$[u_q]$)$\}$
　　$p \leftarrow \tilde{p}$；　$q \leftarrow \tilde{q}$；　$E_{\text{left}}--$；**跳出**
　　如果（$u_q,\ u_p$）$\in E$ **且**（i2o$[u_q]$, i2o$[u_p]$）$\notin E'$ **则**
　　w += 移动(a) 移动$_s$(b) c；　$E' \leftarrow E' \cup \{$(i2o$[u_q]$, i2o$[u_p]$)$\}$
　　$p \leftarrow \tilde{p}$；　$q \leftarrow \tilde{q}$；　$E_{\text{left}}--$；**跳出**
返回 w

定义2.6（规范字符串）。 带标签有向无环图 D 的规范字符串 为:

$$w_D^* \;=\; \text{lexmin}\Big\{w \in \arg\min_{w' \in \mathcal{W}(D)} |w'|\Big\},$$

即，在对 Σ_{SR} 采用固定全序的条件下， $\mathcal{W}(D)$ 中所有最短字符串里字典序最小的元素。

六分量结构元组。 为在保持不变性性质的同时剪枝指数级回溯搜索，我们为每个节点关联一个有向结构描述符。

定义 2.7（六分量结构元组）。 对于带标签有向无环图 D中的节点 v，定义有向的 k 跳邻域:

$$\text{out}_{N_k}(v) = \{u \in V : \text{shortest directed path } v \rightsquigarrow u \text{ has length exactly } k\},$$
$$\text{in}_{N_k}(v) = \{u \in V : \text{shortest directed path } u \rightsquigarrow v \text{ has length exactly } k\}.$$

该六分量结构元组 v 为:

$$\tau(v) = \big(|\text{in}_{N_1}(v)|,\ |\text{out}_{N_1}(v)|,\ |\text{in}_{N_2}(v)|,\ |\text{out}_{N_2}(v)|,\ |\text{in}_{N_3}(v)|,\ |\text{out}_{N_3}(v)|\big) \in \mathbb{N}^6.$$

所有分量　　均通过在每个方向上执行深度至 3 的截断广度优先搜索计算，每个节点的计算成本为 $O(|V|+|E|)$　.

定义 2.8（剪枝规范字符串）。 w_D^{**}剪枝规范字符串的计算方式与 w_D^* 所采用的回溯搜索相同，但在每个 Vℓ/vℓ 分支点还需遵守两项限制。

限制 1（第一操作数资格）。当作用指针节点为 u，且候选项是一个 POW 节点 c，其在 $\sigma(c)$ 中已记录至少一个入邻居时，只有当通过 u 插入 c 会使其成为$\sigma(c)[0]$——即底数时， c 才有资格进行 Vℓ/vℓ 插入。形式上，如果 $\sigma(c)$ 非空且 $\sigma(c)[0] \neq u$，则排除 c。这确保每个 POW 节点都是通过其指定的底数源创建的。

限制 2（基于标签的 τ剪枝）。应用限制 1 后，合格候选集为 $\mathcal{C}$，候选项按标签划分： $\mathcal{C}_{\ell_0} = \{c \in \mathcal{C} :\ \ell$（$c$）$= \ell_0\}$ 对每个不同的标签 ℓ_0，仅保留最大元组候选项:

$$\mathcal{C}^*_{\ell_0} = \big\{c \in \mathcal{C}_{\ell_0} : \tau(c) = \max_{c' \in \mathcal{C}_{\ell_0}} \tau(c')\big\}.$$

已探索的候选集为 $\mathcal{C}^* = \bigcup_{\ell_0} \mathcal{C}^*_{\ell_0}$。选择准则（字符串最短，其次按字典序最小）应用于 $\mathcal{C}^*$ 中的所有分支。

完整算法见表4，采用原地状态变更，并在回溯时显式撤销。

表4：剪枝规范字符串算法的伪代码。PCSSTEP 以递归方式调用，并对 D'、L 和索引映射进行原地变更，在每次回溯时显式撤销。eligible(u, c) 成立，当且仅当 c 不是一个已有第一个操作数且该操作数不等于 u 的二元非交换节点。

算法 PrunedCanonical$(D,\ m)$

输入： 带标签有向无环图 $D=(V,\ E,\ \ell,\ \delta,\ \sigma)$；变量数量 m
输出： 剪枝规范字符串 w_D^{**}
$D \leftarrow$ 规范化_常量_创建(D) // 将所有CONST 创建边重定向到 x_1
预先计算所有 $v \in V$ 的 $\tau(v)$ // $O(|V|(|V|+|E|))$
按 D2S 中的方式初始化 D'、L、i2o、o2i、p、q
返回 PCS步骤$(D, D', L, p, q,$ i2o, o2i $|V|-m, |E|, \varepsilon, \tau)$

函数 PCS步骤$(D, D', L, p, q,$ i2o, o2i $N_{\text{left}}, E_{\text{left}},$ 前缀 $\tau)$:
  如果 $N_{\text{left}}=0$ **且** $E_{\text{left}}=0$ **则返回** 前缀
  最佳 $\leftarrow \perp$
  对于每个$(a,\ b) \in \mathcal{P}_{|V'|}$ **执行**
    $\tilde{p} \leftarrow$ 遍历$(L,\ p,\ a)$; $v_p \leftarrow$ o2i[值$(L, \tilde{p})$]
    如果 $N_{\text{left}} > 0$**那么**
      $\mathcal{C} \leftarrow \{v : (v_p, v) \in E,\ v \notin$ i2o，符合条件$(v_p, v)\}$ // *B9*：第一操作数过滤
      如果 $\mathcal{C} \neq \emptyset$**那么**
        // 标签感知剪枝
        遍历标签 ℓ_0 属于 $\{\ell(c) :\ c \in \mathcal{C}\}$ **执行**
          $G \leftarrow \{c \in \mathcal{C} :\ \ell\ (c)\ = \ell_0\}$
          $\mathcal{C}^*_{\ell_0} \leftarrow \{c \in G :\ \tau(c)\ =$最大值$_{c' \in G}\ \tau(c')\}$
        $\mathcal{C}^* \leftarrow \bigcup_{\ell_0} \mathcal{C}^*_{\ell_0}$
        对每个 $c \in \mathcal{C}^*$ **执行**
          前向：将add c 加入 D' 和循环双向链表；更新映射
          $r \leftarrow$ PCS步骤$(\ldots, pfx\ +$移动$(a)\ +\vee\ell_c, \ldots, N_{\text{left}}-1,\ E_{\text{left}}-1, \ldots)$
          如果 最佳$=\perp$ **或** $(|r|, r)\ <$ (|最佳|, 最佳) **那么** 最佳 $\leftarrow r$
          撤销：从 D'、循环双向链表和索引映射中移除 c
        返回 最佳
      （关于 $v\ell$ 的对称块，使用 q、位移 b，并使用相同的标签感知筛选器）
      （确定性的 *C/c* 块，与 *D2S* 相同；不进行分支）

2.5 规范字符串作为图不变量

现在我们证明 ISALSR 的核心理论结果。关键步骤是定义带标签有向无环图同构，以准确刻画两个表达式有向无环图之间的语义等价性，其中包括非交换运算的操作数顺序约定。

定义 2.9（带标签有向无环图同构）。两个带标签有向无环图 $D_1 = (V_1,\ E_1,\ \ell_1,\ \delta_1)$ 和 $D_2 = (V_2,\ E_2,\ \ell_2,\ \delta_2)$同构，记作$D_1 \cong D_2$，当且仅当存在一个双射 $\phi :\ V_1 \to V_2$，满足：

(i) 边保持： $(u, v) \in E_1 \Leftrightarrow (\phi(u)\ \phi(v)) \in E_2$；**(ii) 标签保持：** $\ell_1(u)\ = \ell_2(\phi(u))$，对所有 $u \in V_1$ 均成立；**(iii) 变量锚定：** 对于每个变量节点 $u \in V_1$，$\delta_1(u)\ = \delta_2(\phi(u))$，即对于所有 $i = 1, \ldots, m$，$\phi(x_i^{(1)})\ = x_i^{(2)}$；**(iv) 操作数顺序保持：** 对于每个具有有序输入列表 $\sigma_1(v)\ = (u_1, u_2)$ 的 POW 节点 $v \in V_1$，

$$\sigma_2(\phi(v)) = (\phi(u_1),\ \phi(u_2)).$$

也就是说，ϕ 将 D_1 中 v 的底数映射到 D_2 中 $\phi(v)$ 的底数，并将指数映射到指数。所有其他运算节点均可交换，不施加顺序约束。

下面给出两个主要结果。它们以猜想的形式陈述，其证明留待未来工作。

猜想 2.10 (往返保真度)。 设 $w \in \Sigma^*_{\mathrm{SR}}$ 含有 $m \geq 1$ 个变量，且 $D = \mathrm{S2D}(w, m)$。若 D 的每个非变量节点都可从某个变量沿有向路径到达，则

$$D \;\cong\; \mathrm{S2D}\big(\mathrm{D2S}(D,\, x_1),\; m\big).$$

猜想 2.11（剪枝规范字符串是完整的带标签有向无环图不变量）。 设 D_1 和 D_2 为带标签有向无环图，且具有 $m \geq 1$ 个变量，二者均满足猜想 *2.10*的可达性条件。那么：

$$w^{**}_{D_1} = w^{**}_{D_2} \quad \Longleftrightarrow \quad D_1 \cong D_2.$$

3 计算实验

猜想 2.10 和 2.11 断言，剪枝规范字符串是完整的带标签有向无环图不变量，但其证明留待未来工作。因此，我们通过检验规范字符串的五项基本性质，对该表示进行实证验证：

P1 往返转换保真度 ty. 将有向无环图编码为字符串，再解码回来，可恢复一个同构的有向无环图 G；$\mathrm{S2D}(\mathrm{D2S}(D), m) \cong D$ (猜想2.10)。

P2 有向无环图无环性。每个由 S2D生成的有向无环图都是无环的（§2.2）。

P3 规范不变性。 剪枝规范字符串是一个完整的同构不变量： $w^{**}_{D_1} = w^{**}_{D_2} \Leftrightarrow D_1 \cong D_2$ (猜想 2.11)。作为推论，规范化是幂等的。

P4 求值保持。 往返转换后的有向无环图在每个测试点上的求值都与原始图相同，误差容限为 10^{-8}。

P5 搜索空间缩减。 规范化将每个表达式的 $\Theta(k!)$ 等价节点编号表示归并为唯一的规范形式（§2.5）。

性质 P1–P4 通过单次随机实验进行验证（§3.3）。性质 P5 则通过单独的搜索空间分析进行验证（§3.4）。可扩展性研究（§3.5）测量剪枝规范化算法的 CPU 时间随有向无环图大小的变化。两项示例性实验（§3.6.1–3.6.2）提供了定性证据，表明规范字符串在表达式有向无环图上诱导出有意义的度量。所有实验均使用全局随机种子 42 ，并可通过公开代码仓库完全复现。结果见第 4节。

3.1 基准测试套件

我们从两个标准符号回归套件中选取基准表达式：Nguyen 基准测试，以及一部分 AI Feynman 方程。

Nguyen 基准测试。 表5 列出了由Uy 等人 [2011]最初提出、并被Liu 等人 [2025]采用的配置中的 12 个 Nguyen 基准表达式。其中 8 个是在有界区间上的单变量多项式、三角函数、对数函数或根式函数；另外 4 个是双变量表达式，涉及乘积、求和以及唯一的非交换算子 POW。训练数据由从指定输入范围内均匀采样的 20 个点组成；测试数据由从同一范围内采样的 100 个点组成。

表 5: Nguyen 符号回归基准测试 准测试，配置遵循 Liu 等人 [2025]。训练：2 0 均匀采样；测试 ：100 个均匀采样；随机种子 42。

Name	表达式 $f(x)$ 或 $f(x,\ y)$	m	输入范围
Nguyen-1	$x^3 + x^2 + x$	1	$[-1, 1]$
Nguyen-2	$x^4 + x^3 + x^2 + x$	1	$[-1, 1]$
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	1	$[-1, 1]$
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1	$[-1, 1]$
Nguyen-5	正弦(x^2) 余弦$(x) - 1$	1	$[-1, 1]$
Nguyen-6	正弦(x) +正弦$(x + x^2)$	1	$[-1, 1]$
Nguyen-7	ln$(x+1)$ +ln(x^2+1)	1	$[0, 2]$
Nguyen-8	$\sqrt{x}$	1	$[0, 4]$
Nguyen-9	正弦(x) +正弦(y^2)	2	$[-1, 1]$
Nguyen-10	2 正弦(x) 余弦(y)	2	$[-1, 1]$
Nguyen-11	x^y	2	$[0, 1]$
Nguyen-12	$x^4 - x^3 + y^2/2 - y$	2	$[-1, 1]$

AI Feynman基准 表 6 列出了从Udrescu 和 Tegmark [2020], 的 AI Feynman 数据集中选取的 10 个物理方程，所选子集和变量范围遵循Liu 等人 [2025]。所选方程包括 1 个单变量方程、5 个双变量方程和 4 个三变量方程，涵盖多项式、三角函数和有理函数形式。数据共包含 200个样本，采用80/20训练/测试划分。

表 6：AI Feynman基准子集，遵循Liu 等人 [2025]的选择和变量范围。数据：200个样本，80/20 训练/测试划分，随机种子42。

ID	表达式	m	变量范围
I.6.20a	$e^{-\theta^2/2}/\sqrt{2\pi}$	1	$\theta \in [1,3]$
I.12.1	μN_s	2	$[1,5]^2$
I.25.13	q/C	2	$[1,3]^2$
I.34.27	$\hbar\omega$	2	$[1,5]^2$
I.39.10	$p_r V/2$	2	$[1,5]^2$
II.3.24	$p\,r/(4\pi)$	2	$[1,5]^2$
I.14.3	$m\,g\,z$	3	$[1,5]^3$
I.12.4	$q_1/(4\pi r c)$	3	$[1,5]^3$
I.10.7	$m_0/\sqrt{1-v^2/c^2}$	3	$m_0{\in}[1,5]$, $v{\in}[1,2]$, $c{\in}[3,10]$
I.48.20	$mc^2/\sqrt{1-(v/c)^2}$	3	$m{\in}[1,5]$, $c{\in}[1,2]$, $v{\in}[3,10]$

3.2 随机表达式生成

实验中使用两种互补的流程生成测试输入。

随机有向无环图生成。 可扩展性（§3.5）和搜索空间（§3.4）实验需要生成内部节点数量预先指定的随机有向无环图。给定 m 个输入变量和目标数量 k 个内部节点，生成器会创建 m 个 VAR 叶节点，然后迭代添加 k 个运算节点。每一步中，以 0.6 的概率从 {SIN,COS, EXP, LOG, ABS} 中均匀选择一个一元操作；否则，从{ ADD,MUL,SUB, DIV} 中选择一个二元操作。如果节点数少于两个，则强制选择一元操作。每个新节点从所有先前创建的节点中均匀随机选取一个（一元）或两个（二元）节点，并接入相应的入边。默认排除 POW，因此根据 §2.1 中的分解，所有二元操作都是可交换的（SUB $\to$ ADD +NEG，DIV $\to$ MUL + INV）。每样本种子是确定性的：种子为 $= 42 + m \cdot 10{,}000 + k \cdot 100 + s$，其中 s 是样本索引。

随机字符串生成。 性质验证实验（§3.3）生成随机 IsalSR 字符串作为输入。随机字符串的生成方式是：先从均匀分布 $n \sim \text{Uniform}(1, T_{\max})$ 中采样词元数量，然后从完整的 IsalSR词元集（定义 2.2）中独立且均匀地抽取每个词元。每个字符串都通过 S2D（表 2）进行解析；解析后仅包含 VAR 节点（没有内部节点）的有向无环图会被丢弃。该策略对字符串空间进行均匀采样，而非针对特定的有向无环图拓扑，从而覆盖所有词元类型，包括移动、边创建和空操作指令。

3.3 性质验证：P1–P4

我们通过单次遍历随机生成的 IsalSR 字符串来验证性质 P1–P4。对于每个 $m \in \{1,2,3\}$ 输入变量，我们生成 $N = 5{,}000$ 个包含 $T_{\max} = 20$ 个词元的随机字符串。每个有效且不只包含 VAR 节点的有向无环图 D均接受以下四项测试：

1. P2（无环性）：对 D 进行拓扑排序；若排序包含全部 $|V|$ 节点，则测试通过。2. P1（往返转换）：将贪心编码 $w' = \text{D2S}(D)$ 解码为 $D' = \text{S2D}(w', m)$；若$D \cong D'$ 符合定义 2.9，则测试通过。3. P4（求值保持）：在每种 m配置下，分别在五个固定测试点上对 D 和往返转换后的 D' 进行求值，测试点涵盖负值、零和正值输入。若所有测试点上的最大绝对误差低于 10^{-8}，则测试通过。当两个有向无环图在某个测试点产生相同的求值误差（例如除以零）时，该点计为保持。4. P3（规范不变性）：将剪枝规范字符串 $w^{**} = \text{canonical}(D)$ 解码为 $D'' = \text{S2D}(w^{**}, m)$；若 $D \cong D''$，则不变性测试通过。随后对 D''再次进行规范化；若所得字符串等于 w^{**}，则幂等性测试通过。规范化超时设为 2.0 秒；超过超时的样本不计入 P3 的分母，但仍计入 P1、P2 和 P4 的统计数。

报告通过率及其 Clopper–Pearson 精确 95% 二项置信区间，随机字符串总数为 $3\times5,\ 000 = 15,\ 000$。

作为补充验证，我们还对从 Nguyen（表 5 ）和 AI Feynman（表 6）基准测试套件中选取的 $m \in \{1, 2, 3\}$ 个基准表达式子集，确定性地验证性质 P1–P4；这些表达式涵盖的输入变量数量和有向无环图规模从 $|V| = 3$ 到 $|V| = 14$。

3.4 性质验证：P5 —搜索空间缩减

本实验直接验证了核心主张：规范化会将每个表达式的 $\Theta(k!)$ 种等价节点编号表示折叠为一种规范形式。

我们没有对随机字符串进行采样并测量碰撞率——这种方法会低估真实的等价类大小，因为字符串空间远大于结构上唯一的有向无环图数量。相反，我们有意构造每个表达式的所有等价表示，并验证规范化会将它们映射到同一个字符串。

带标签有向无环图 $D = (V, E, \ell, \delta)$ 有 m 个变量节点（标识符为 0、.....、$m-1$）和 k 个内部节点（标识符为 m、.....、$m+k-1$），共有 $k!$ 种不同的节点编号方案：对内部节点 ID 进行排列，从而生成结构不同但编码同一数学表达式的带标签有向无环图。如果规范字符串是完整的同构不变量，那么它必须将所有 $k!$ 种编号方案映射到同一个字符串。

对于 $k \in \{1, \ldots, 12\}$ 个 $m \in \{1, 2\}$ 输入变量的每个取值，我们都会生成结构唯一的随机有向无环图（§3.2），并按规范字符串去重，以确保每个有向无环图都表示一个不同的同构类。对于每个具有 D 个 k 内部节点的有向无环图，我们按如下步骤进行：

1. 计算一次剪枝规范字符串 $w^{**} = \text{canonical}(D)$。2. 生成排列：若 $k \le 8$ ($k! \le 40{,}320$)，则生成全部 $k!$ 种排列；否则生成 50,000 个随机排列。3. 对于 $\{0, 1, \ldots, k-1\}$ 的每个排列 π，通过将内部节点 $m+i$ 重新映射到位置 $m+\pi(i)$ 来构造 D_π，同时保留所有边、标签和操作数顺序。计算结构指纹：按新 ID 顺序，为每个节点 v 生成元组 $\big(\ell(v)\ \ \sigma(v)\big)$，其中 $\sigma(v)$ 是有序输入列表（定义 2.1）。4. 在 100 个排列的子集上验证规范不变性：计算 canonical(D_π)，并检查其是否等于w^{**}。

两个不同的排列 π_1, π_2 当且仅当它们相差一个 D 的自同构时，才会产生相同的结构指纹。根据轨道-稳定子定理，不同结构指纹的数量恰好为

$$n_{\text{distinct}} = \frac{k!}{|\mathrm{Aut}(D)|}, \tag{2}$$

其中 $|\mathrm{Aut}(D)|$ 是 D 的自同构群的大小。对于没有对称性的通用有向无环图（$|\mathrm{Aut}(D)| = 1$），该值等于 $k!$ 因此，归一化比率 $n_{\text{distinct}}/k!$ 衡量了每个有向无环图的结构非对称性，我们会将其与规范不变性成功率一并报告。

3.5 规范化可扩展性

本实验测量剪枝规范化算法（定义 2.8）的实际运行时间如何随有向无环图大小变化，并量化 τ 元组剪枝相较于穷举搜索（定义 2.6）所带来的加速效果。

我们采用全因子设计，针对 $k \in \{1, 2, \ldots, 15\}$ 个内部节点和 $m \in \{1, 2, 3\}$ 个输入变量生成随机有向无环图；每种（k, m）配置使用 200 个独立设定随机种子的样本，总计生成 $15 \times 3 \times 200 = 9{,}000$ 个有向无环图。每个有向无环图都进行两次规范化：一次使用穷举回溯算法，另一次使用剪枝算法。每次规范化的超时时间均设为 120 秒。实际运行时间通过单调高精度计数器测量。

由于规范化耗时呈现明显的右偏分布，我们报告中位数和四分位距（IQR），而不是均值。加速比 定义为每种 (k, m) 配置下穷举耗时的中位数除以剪枝耗时的中位数。

3.6 度量空间性质

如果剪枝规范字符串是完备不变量（猜想 2.11），那么规范字符串上的莱文斯坦编辑距离 [Levenshtein,1966] 满足不可分辨者同一性，并在表达式有向无环图的同构类上定义了一种度量。我们通过两个示例性实验来探究这种度量。

3.6.1 表达式有向无环图之间的最短路径距离

我们选取了五组表达式对，涵盖一系列结构关系：(A) 结构不同的表达式（正弦(x) 与 x^2+x），(B) 操作数结构相同但二元组合算子不同（正弦(x) + 余弦(x) 与 正弦(x) · 余弦(x)），(C) 存在包含关系的表达式（x^2 与 x^3+x^2+x），(D) 拓扑相同但一元算子不同（exp(x) 与 对数(x)），以及 (E) 混合双变量表达式（正弦(x) + y^2 与 余弦(x) · y）。对于每组表达式对，我们都将两个表达式转换为带标签的有向无环图，计算其剪枝规范字符串 w^{**}，并使用 Wagner–Fischer 动态规划算法及完整的编辑操作回溯求得莱文斯坦距离。本实验仅作示例，不涉及随机化或统计分析。

3.6.2 距离为 1 的邻域分析

本实验量化了规范化在规范字符串的直接邻域中消除的冗余。我们取表达式 sin(x_1) +cos(x_1)，其中 $m=1$ 为变量，其剪枝规范字符串为w^{**} = VcVspv+Ppc，长度为 $L=10$。

我们枚举 w^{**} 的所有字符级莱文斯坦距离为 1 的邻居。令 $|\mathcal{A}|$ 表示字符级字母表大小（此处为 $|\mathcal{A}|=17$，包含 7 个单字符指令词元，以及 10 个作为复合词元第二个字符出现的标签字符）。邻居总数为：

$$N_{\text{total}} = \underbrace{L}_{\text{deletions}} + \underbrace{L\cdot(|\mathcal{A}|-1)}_{\text{substitutions}} + \underbrace{(L+1)\cdot|\mathcal{A}|}_{\text{insertions}} = 10+160+187 = 357. \tag{3}$$

每个邻居字符串均通过 S2D 解析；无法解析或生成仅含 VAR 的有向无环图的字符串均归类为无效。有效有向无环图会进行规范化（超时 5 秒），并统计不同规范形式的数量。冗余率 定义为 $1-N_{\text{unique}}/N_{\text{valid}}$，其中 N_{unique} 是 N_{valid} 个有效邻居中不同规范字符串的数量。结果还会按编辑操作类型（删除、替换、插入）进一步细分。本实验是针对单个表达式的案例研究。

3.7 统计方法与计算基础设施

统计方法。 性质验证实验（§3.3）报告每个通过率的 Clopper–Pearson 精确95% 二项分布置信区间，该区间具有保守性，且对所有样本量均有效。搜索空间实验（§3.4）报告穷举排列运行的精确计数（$k\le 8$）以及抽样运行的下界计数（$k>8$）；规范不变性成功率是精确的二项比例。可扩展性实验（§3.5）报告中位数和四分位距，这些统计量对于执行时间测量中常见的右偏分布具有稳健性。所有实验均使用底数随机种子 42，每样本种子由底数种子、变量数量、 DAG 大小和样本索引确定性地派生。

计算基础设施。 两个示例性实验（§3.6.1–3.6.2）是确定性的单表达式计算，在标准工作站上运行时间不到一分钟。其余实验在马拉加大学超级计算与生物创新中心（SCBI）的 Picasso 超级计算机上运行，使用仅配备 CPU 的节点。表 7 总结了每个实验的参数和计算规模。

表 7：实验参数汇总。 k：内部节点； m：输入变量； N：每种配置的样本数。除两个示例性实验外，所有实验均在 Picasso 超级计算机上运行。

	实验	k	m	N	超时	总计
P1–P4	性质验证	—	1–3	5,000/m	2 s	15,000
P5	搜索空间	1–12	1–2	∼40/config	5 s	961
	可扩展性	1–15	1–3	200/配置	120 秒	9,000
	最短路径	—	1–2	5 对	—	5
	邻域	—	1	357 个邻居	5 s	357

4 结果

我们展示第3节所述实验的结果。首先报告性质验证结果（P1–P4）和搜索空间缩减分析（P5），然后报告可扩展性测量，最后介绍两个度量空间案例研究。

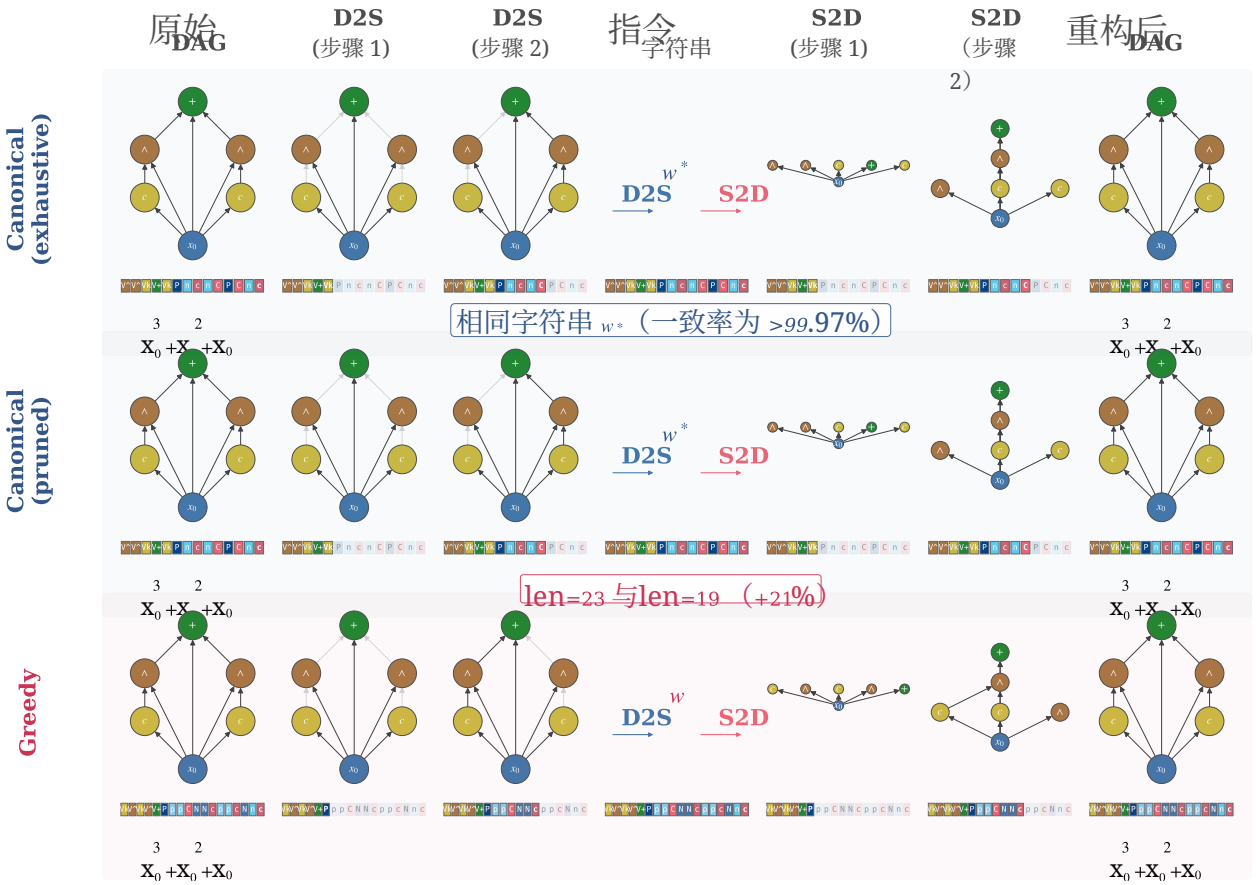


图3：Nguyen-1基准测试的往返转换性质（$x_0^3 + x_0^2 + x_0$）。每一行应用一种不同的有向无环图到字符串（D2S）算法，随后通过字符串到有向无环图（S2D）进行重建。**第0、6列**：原始有向无环图和重建后的有向无环图（同构，$\cong$）。**第1–2列**：D2S逐步对有向无环图进行编码；幽灵节点（虚线）和边表示尚未编码的部分，而指令字符串则从左向右逐渐生成。**第3列**：完整的指令字符串（规范算法为w^*，贪心算法为 w）。**第4–5列**：S2D根据字符串重建有向无环图。规范算法和剪枝算法生成相同的字符串（$|w^*|$=19，>99.97%的一致率），而贪心算法生成的编码更长（$|w|$=23，+21%），但三种算法往返转换后都得到同构的有向无环图。

4.1 性质验证（P1–P4）

我们报告第 §3.3节所述随机化性质验证实验的结果。在生成的 15,000 个随机字符串（每个 5,000 个 $m \in \{1, 2, 3\}$）中，14,841 个生成了至少包含一个内部节点的有效有向无环图，并构成性质 P1、P2 和 P4 的测试样本池。对于 P3，5,713 个样本超过了 2.0 秒的规范化超时，因此未计入分母，剩余 9,128 个可评估样本。

P1（往返保真度）。所有14,841 个样本均通过：S2D(D2S(D) m) $\cong D$ 在所有情况下均如此。真实通过率的 Clopper–Pearson95% 置信区间为 [99.975%, 100%]。图3 展示了 Nguyen-1 基准测试的往返转换，其中规范算法和剪枝算法生成相同的字符串（$|w^{**}| = 19$），而贪心编码器生成的字符串更长（$|w| = 23$，开销为21%）。

P2（有向无环图无环性）。所有14,841 个解码后的有向无环图均无环，置信区间相同：[99.975%, 100%]。P2 由构造保证：S2D 为新节点分配严格递增的标识符，并拒绝可能生成环的 c/c 边（§2.2）。实验确认该实现符合此规范。

P4（求值保持）。全部 14,841 个往返转换后的有向无环图在每个测试点上的求值均与其原始版本完全一致（最大绝对误差低于 10^{-8}），置信区间为 [99.975%，100%]。如果原始有向无环图和往返转换后的有向无环图产生相同的求值误差（例如除以零），则计为保持。

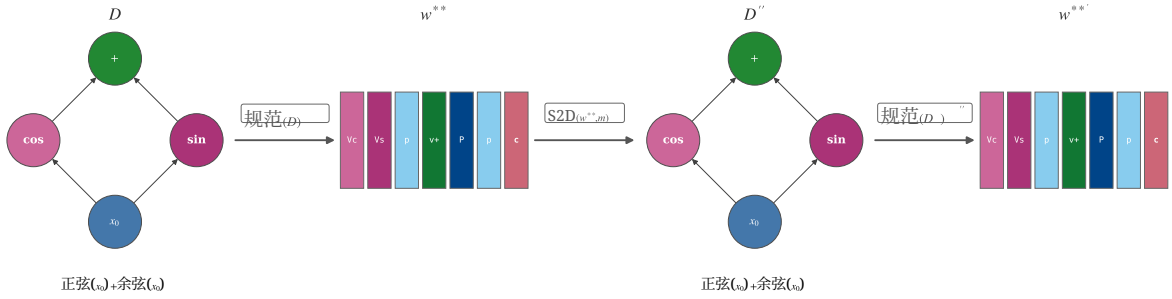


图 4：对于 $\sin(x_0) + \cos(x_0)$ 的性质 P3（规范不变性和幂等性）。原始有向无环图 D 经规范化后得到 $w^{**} = \text{VcVspv+Ppc}$，再通过 S2D 解码以重构 D''，其结构与 D 同构（**不变性**：$D \cong D''$）。再次进行规范化得到 $w^{**'} = w^{**}$（**幂等性**），由此确认规范字符串是合成与重新规范化映射的不动点。颜色编码的词元块展示了每个转换步骤中的指令字符串。

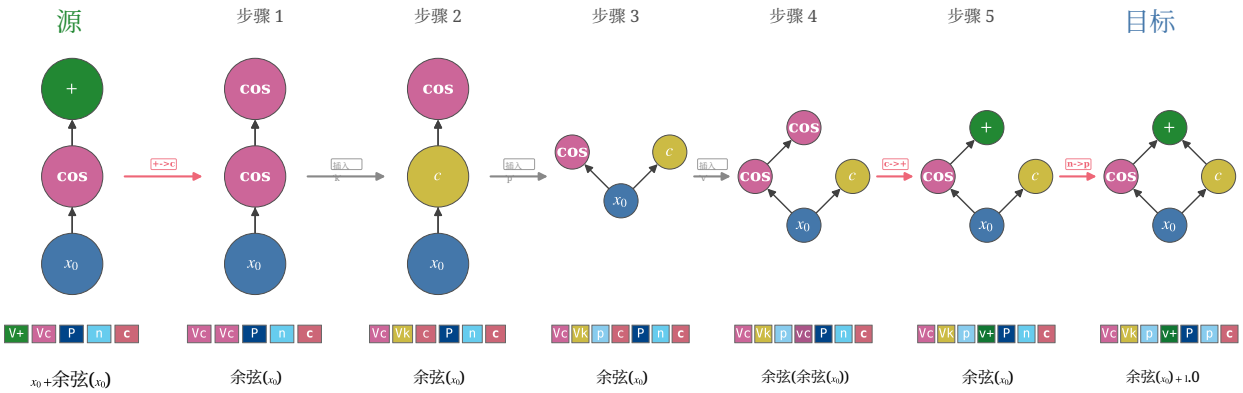


图 5：规范字符串空间（$d_{\text{Lev}} = 6$）中 $\cos(x_0)+x_0$ (V+VcPnc) 与 $\cos(x_0)+1$(VcVkpv+Ppc) 之间的最短 Levenshtein 路径。每个步骤都对当前字符串进行一次字符级编辑（替换或插入）。所有中间字符串都会生成有效的表达式有向无环图，路径经过简化的 $\cos(x_0)$ 形式——包括步骤 4 中绕经 $\cos(\cos(x_0))$——最终到达目标。每个有向无环图下方是对应的 IsalSR 指令字符串（按词元类型进行颜色编码）和数学表达式。

基准测试验证。表 8 报告了对 8 个来自 Nguyen 和 AI Feynman 套件的基准表达式进行 P1–P4 确定性验证的结果。对于每个表达式，四项性质均成立，有向无环图规模范围从 $|V| = 3$（Nguyen-8，$\sqrt{x}$）到 $|V| = 14$（Nguyen-12，一个包含两个变量的四项多项式）。

表 8：对基准表达式进行 P1–P4 的确定性验证。$|V|$：节点数；$|E|$：边数；$|w^{**}|$：剪枝后规范字符串长度。

表达式	m	$\lvert V\rvert$	$\lvert E\rvert$	深度	P1	P2	P3	P4	$\lvert w^{**}\rvert$
Nguyen-1	1	6	9	3	✓	✓	✓	✓	19
Nguyen-5	1	8	10	5	✓	✓	✓	✓	26
Nguyen-7	1	9	12	5	✓	✓	✓	✓	32
Nguyen-8	1	3	3	2	✓	✓	✓	✓	7
Nguyen-9	2	7	7	4	✓	✓	✓	✓	19
Nguyen-10	2	6	6	2	✓	✓	✓	✓	16
Nguyen-12	2	14	21	4	✓	✓	✓	✓	56
Feynman-I.14.3	3	4	3	1	✓	✓	✓	✓	7

P3（规范不变性）。在 9,128 个未超时样本中，9,125 个通过（99.97%）。表 9报告了逐项m细分。

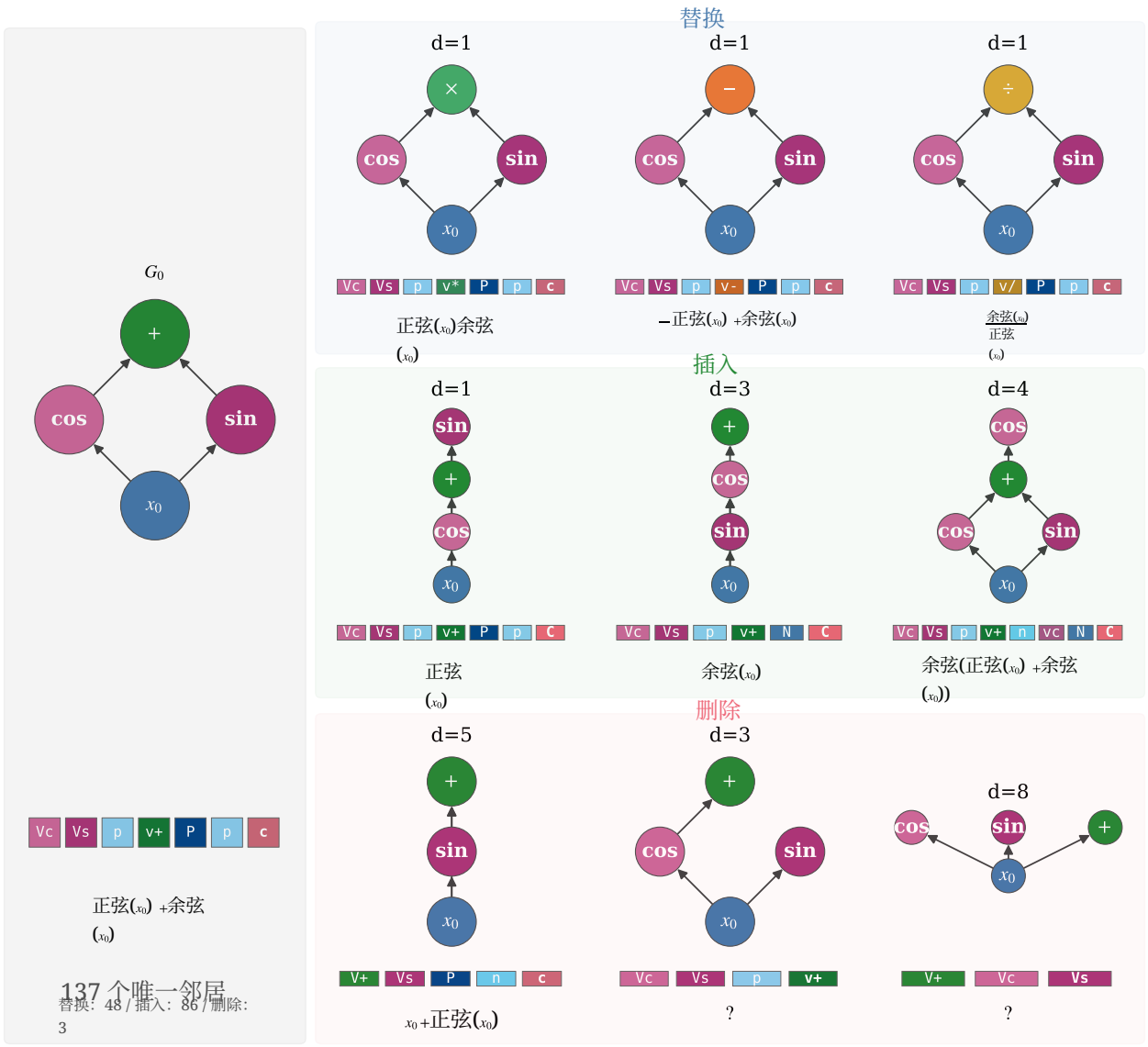


图6： G_0 周围规范字符串空间中的邻域结构：sin(x_0) +cos(x_0)（规范字符串 VcVspv+Ppc）。每个邻居都是通过对规范字符串执行一次莱文斯坦编辑（替换、插入或删除），然后重新规范化得到的。 d 标签表示邻居的规范字符串与底数之间的莱文斯坦距离。**替换** 邻居（d=1）只改变组合运算（$+\to\times$、 $+\to-$、 $+\to\div$），保留操作数结构。**插入** 和**删除** 邻居会产生结构不同、距离各异的有向无环图。在132个有效的 Lev-1 扰动中，只有38个产生不同的规范形式（冗余率为71.2%），这说明了 $\Theta(k!)$ 搜索空间缩减。

三个样本未通过 P3： 样本索引 274，在 $k = 14$ 用于 $m \in \{2, 3\}$；样本索引 2,603，在 $k = 6$ 用于 $m = 3$。非 e
这些故障是超 时导致的。我们在本地重现了全部三种故障，并找到了一个共同的根本原因。

P3 失败的根本原因。在这三个案例中，失败的有向无环图都包含一条指向某个VAR 节点的边——当 C/c 标记将目标指向 CDLL 上的变量位置时，S2D 可能产生这种结构。具体而言，在样本 2603 ($m = 3, k = 6$) 中，一个 CONST 节点（标识符 6）有一条边指向 VAR 节点 x_0（标识符 0）：该有向无环图包含边（6，0）。同样，样本 274 包含一条从 CONST 节点指向 x_0 at $k = 14$ 的边。这些边在语义上是无效的——VAR 节点在求值过程中会忽略其输入，这解释了为何 P4（求值保持）能够通过——但它们构成了规范 D2S 算法无法编码的结构。当剩余的未编码边中包含一条指向

表9：按输入变量数量 m 划分的 P3 通过率。Clopper–Pearson 精确 95% 置信区间。

m	已评估	Pass	Rate	95% 置信区间
1	3,004	3,004	100%	[99.88%, 100%]
2	3,062	3,061	99.97%	[99.82%, 100%]
3	3,062	3,060	99.93%	[99.74%, 100%]
All	9,128	9,125	99.97%	[99.89%, 100%]

叶节点的边时，没有候选节点满足插入条件，因此该算法以运行时错误终止，而不是超时。穷举规范算法和剪枝规范算法在这些有向无环图上均以相同方式失败，证实r-元组剪枝启发式并非问题所在。
贪心 D2S 能正确处理这些有向无环图（P1 通过）：两个端点都进入循环双向链表后，它会用 C/c 标记编码伪边。然而，规范回溯搜索会假定所有边都终止于运算节点，并据此跟踪剩余边；指向 VAR 节点的边违反了这一不变条件。

这些失败不会影响猜想 2.11。该猜想要求两个有向无环图都满足猜想 2.10 的可达性条件：“每个非变量节点都能从某个变量沿有向路径到达。”尽管这三个失败的有向无环图在技术上满足这一条件（指向 VAR 节点的边不会妨碍到达运算节点），但它们包含一种退化结构——指向叶节点的输入——不属于 D2S 的设计适用范围。在规范化之前移除 VAR 节点入边的规范化步骤，可以彻底消除这些失败。

图 4 展示了 $\sin(x_0)+\cos(x_0)$的规范不变性和幂等性。这是一个规范化成功的良构有向无环图：规范字符串 $w^{**} = \texttt{VcVspv+Ppc}$ 是合成与重新规范化映射的不动点。

总结。 在全部 14,841 个有效样本中，性质 P1、P2 和 P4 均无一例外地成立。对于有向无环图中不含指向变量节点的边的 100% 9,125 个未超时样本，P3 均成立。P3 的三次失败均由一种单一的退化有向无环图结构导致——一条指向变量节点的边——而规范 D2S无法对其编码。该结构在语义上毫无意义，可以通过一个规范化步骤消除。这些失败并不与猜想 2.10或 2.11相矛盾。

4.2 搜索空间缩减（P5）

我们报告 §3.4 中所述受控置换实验的结果。我们分析了总计 961 个结构唯一的随机有向无环图，涵盖 $k \in \{1,\dots,12\}$ 个内部节点和 $m \in \{1,2\}$ 个输入变量。对于 $k \le 8$，穷举了全部 $k!$ 种排列（721 个有向无环图）；对于 $k \in \{9,\dots,12\}$，每个有向无环图随机抽取 50,000 种排列（共 240个有向无环图）。

不同表示。图 7(a) 在对数刻度上绘出了每个有向无环图的不同结构指纹数量随 k 的变化，并与理论上的 $k!$ 曲线并列展示。在穷举范围内（$k \le 8$），箱线图在跨越五个数量级的范围内均落在 $k!$ 线上 —— 从 $k! = 1$在 $k = 1$ 到 $k! = 40{,}320$ 在 $k = 8$——这证实了等价类大小等于 $k!/|\mathrm{Aut}(D)|$（公式 2）。在 $k = 8$ 处，中位数有向无环图有完整的 40,320 种不同表示，印证了第 1 节所述的冗余规模。对于 $k > 8$，采样计数是不同表示真实数量的下界，但仍持续跟随 $k!$ 的增长率。规范化之后，所有等价表示都归并为一个规范字符串（图 7(a) 中 $y = 1$ 处的红线）；数据与这条线之间的阴影区域表示规范化所消除的冗余。

自同构结构。 图 7(b) 展示了穷举范围内的归一化比率 $n_{\text{distinct}}/k!$。大多数有向无环图都达到完整的 $k!$ 上界（比率为 $= 1.0$），对应于自同构群平凡的通用表达式（$|\mathrm{Aut}(D)| = 1$）。异常值出现在比率 $= 0.5$ ($|\mathrm{Aut}(D)| = 2$)、0.25（$|\mathrm{Aut}(D)| = 4$）和 ≈ 0.17 ($|\mathrm{Aut}(D)| = 6$) 处。这些对应于具有非平凡对称性的有向无环图，例如，一个可交换的二进制节点（ADD 或 MUL）的两个子树同构，从而得到 $|\mathrm{Aut}(D)| = 2$。对于每个经过穷举验证的有向无环图，观测到的比率都与轨道-稳定子定理的预测完全吻合。

规范不变性。 对于每个有向无环图，我们通过重新计算$\mathrm{canonical}(D_\pi)$ 并检查其是否与 w^{**} (§3.4，步骤 4) 相等，验证了961 个有向无环图中的100%规范不变性：没有任何排列产生不同于原始字符串的规范字符串。未观察到失败或超时。

总结。 置换分析证实，包含 k 个内部节点的每个表达式都对应 $k!/|\mathrm{Aut}(D)|$ 种结构上不同的节点编号表示，而剪枝规范字符串会将它们全部归并为唯一形式。对于一般有向无环图，理论上的 $\Theta(k!)$ 上界是紧确的，且这种约简是精确的：在测试的 961 个有向无环图中，规范化消除了所有等价表示。

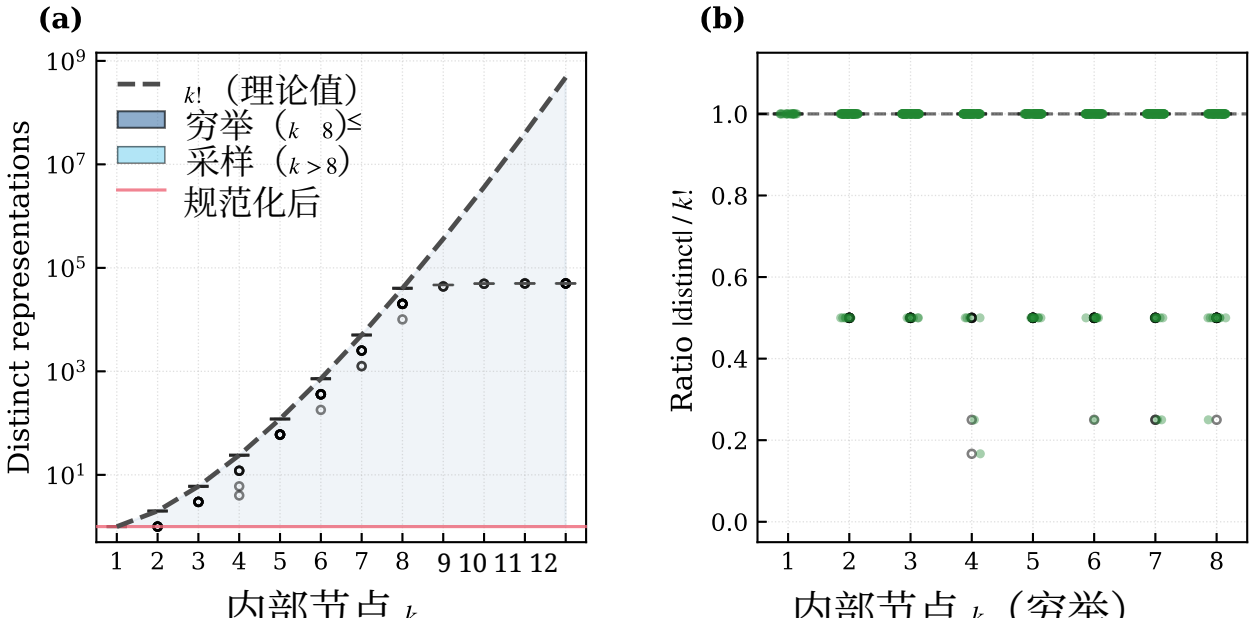


图 7： $\Theta(k!)$ 搜索空间缩减的实证验证。对于 961 个包含 k 个内部节点（$m \in \{1, 2\}$ 个变量）的随机表达式有向无环图，应用了内部节点 ID 的全部 $k!$ 种排列（对 $k \leq 8$ 进行穷举；对 $k > 8$ 抽取 50,000 个样本），生成结构不同但编码相同数学表达式的带标签有向无环图。**(a)** 每个表达式的不同表示数量（对自同构取模的节点编号），与理论上的 $k!$ 曲线（虚线）进行比较。对于 $k \leq 8$，穷举箱线图在跨越五个数量级的范围内均落在 $k!$ 线上，证实等价类大小等于 $k!/|\mathrm{Aut}(D)|$。对于 $k > 8$，抽样计数是下界。位于 $y = 1$ 处的红色水平线标示规范化后的结果：所有等价表示均归并为单个规范字符串（阴影区域 =表示冗余已消除）。**(b)**在穷举范围（$k \leq 8$）内，测得的不同表示数量与 $k!$ 的归一化比率。大多数有向无环图达到完整的 $k!$ 上界（比率为 $= 1.0$）；低于 1.0 的离群值对应具有非平凡自同构群的有向无环图，此时根据轨道-稳定子定理，$n_{\text{distinct}} = k!/|\mathrm{Aut}(D)|$。在全部 961 个有向无环图中，规范不变性的验证率为 100%。

4.3 规范化可扩展性

§3.5 中所述的可扩展性实验已在 Picasso 超级计算机上执行。结果仍在分析中，将在后续修订版中报告。

4.4 度量空间性质

若猜想 2.11 成立，则剪枝规范字符串上的莱文斯坦距离 [Levenshtein、 1966] 满足不可分辨者同一性，因此在表达式有向无环图的同构类上定义了一个度量。我们通过两个实验探究该度量：计算选定表达式对之间的最短路径距离（§4.4.1），以及围绕单个表达式进行距离为 1 的邻域分析（§4.4.2）。

4.4.1 最短路径距离

表 10 报告了五组表达式对的剪枝规范字符串之间的莱文斯坦距离。这些表达式对经过挑选，以涵盖一系列结构关系（§3.6.1）。

这些距离与配对表达式之间的结构相似性相吻合。配对 B 和 D 的结果均为 $d_{\text{Lev}} = 1$，对应于规范字符串中的单字符替换：配对 B 的组合子标签 + → *，以及配对 D 的一元标签 e → l。两种情况下，有向无环图拓扑都相同，只有一个节点的操作类型不同，编码将其表示为单字符变化。

表 10：五组表达式对的剪枝后的规范字符串之间的莱文斯坦距离。每组表达式对旨在隔离一种特定的结构关系。

Pair	表达式 1	表达式 2	关系	d_{Lev}
A	正弦(x)	$x^2 + x$	结构不同	10
B	正弦(x) + 余弦(x)	正弦(x) · 余弦(x)	操作数相同，组合子不同	1
C	x^2	$x^3 + x^2 + x$	包含关系	12
D	指数函数(x)	对数(x)	相同拓扑，不同一元运算符	1
E	正弦(x) + y^2	余弦(x) · y	混合双变量	12

配对 A（$d_{\text{Lev}} = 10$）和配对 C（$d_{\text{Lev}} = 12$）涉及的表达式在拓扑和大小上都不同，需要通过多次插入、删除和替换操作，才能将一个规范字符串转换为另一个。配对 C 尤其具有说明性：尽管 x^2 在结构上包含于 $x^3 + x^2 + x$ 中，但这两个表达式的规范字符串没有任何共同前缀，因此编辑距离反映的是完整的结构差异，而非简单的扩展。

配对 E ($d_{\text{Lev}} = 12$) 在共享的双变量设置（$m = 2$）中结合了运算符类型、有向无环图拓扑和变量使用方面的差异。

图5 展示了另一个独立示例：在余弦(x_0)+x_0 与余弦(x_0)+1($d_{\text{Lev}} = 6$)之间逐步经过的 Levenshtein 路径。之所以选择此例，是因为其中间距离产生了一条非平凡但在视觉上易于追踪的路径。每个中间字符串都会生成一个有效的表达式有向无环图。这条路径经过了结构上截然不同的中间形式，包括一段余弦(余弦(x_0)) 绕行，这说明编辑距离并不总是遵循单调的结构插值。

4.4.2 距离为1的邻域

我们按照 §3.6.2中的规程，分析正弦(x_0)+余弦(x_0) 的距离为1的邻域（规范字符串 $w^{**} = \texttt{VcVspv+Ppc}$，长度为 $L = 10$，字符级字母表大小为 $|\mathcal{A}| = 17$）。整个邻域共包含357个字符串：10 次删除、160 次替换操作和187 次插入（方程3）。

Table 11 按编辑操作汇总结果

类型。

表11：sin(x_0) +cos(x_0) (w^{**} = VcVspv+Ppc) 的距离为1的邻域，按编辑操作细分。N_{total}：Lev-1 字符串的数量； N_{valid}：可生成有效非平凡有向无环图的字符串； N_{unique}：有效字符串中的不同规范形式。

操作	N_{total}	N_{valid}	N_{unique}	冗余
删除	10	5	3	40.0%
替换	160	60	35	41.7%
插入	187	67	18	73.1%
All	357	132	38	71.2%

在 357 个邻居中，132 （37.0%）可生成至少包含一个内部节点的有效有向无环图。其余 225 个字符串要么无法解析为有效有向无环图，要么生成平凡的仅含变量的图。在这 132 个有效邻居中，只有 38个生成彼此不同且不同于原始字符串的规范字符串，因此总体冗余率为 71.2%。表 11 中按操作类型统计的唯一数量之和超过 38 ，因为有些规范形式可以通过不止一种编辑类型得到。另外还有 34 个有效字符串规范化后会变回原始字符串 w^{**}，这意味着相应的编辑被规范化过程“吸收”，不会改变所表示的表达式。

三种编辑操作呈现出不同的冗余特征。删除操作的绝对数量最低（10 个中有 5 个有效），但冗余程度中等（40.0%），因为从较短的字符串中删除单个字符往往会彻底破坏有向无环图结构。替换操作产生的有效邻居绝对数量最多（60 个），冗余程度也适中（41.7%）：许多替换只改变复合词元的标签字符，从而映射到不同的操作类型，进而得到不同的规范形式。插入操作生成的候选项总数最多（187 个），冗余率也最高（73.1%）：许多插入的字符会生成移动词元或空操作，而规范化会将其丢弃，使生成的有向无环图最终归并为少数几种规范形式。

图6 展示了邻域，其中 d 标签表示每个邻居的规范字符串与原始规范字符串之间的莱文斯坦距离。位于 $d = 1$ 的替换邻居对应于单个标签的更改（例如 $+ \to \times$，$+ \to -$），同时保留操作数结构；而位于 $d > 1$的插入和删除邻居会产生结构上

不同的有向无环图。较高的冗余率体现了规范化带来的搜索空间压缩：近四分之三的有效扰动都会归并为先前出现过的规范形式。

5 结论

我们提出了 ISALSR，这是一种表示框架，可将符号回归表达式编码为双层字母表上的指令字符串，并计算剪枝规范字符串，作为带标签表达式有向无环图的完备同构不变量。该框架解决了所有现有符号回归方法固有的结构冗余问题：对于一个具有 k 个内部节点的有向无环图，有 $\Theta(k!)$ 种不同的节点编号方案会编码同一个表达式，并消耗搜索预算，却无法带来多样性。ISALSR 将这种冗余归并为每个同构类对应的单一规范形式，并可作为预处理步骤集成到任何现有符号回归算法中，只需在适应度评估之前调用规范化过程。

主要技术贡献有三点。第一，双层字母表确保字母表上的每个字符串都能解码为有效有向无环图，从而无需使用基于语法或基于类型系统的有效性过滤器通过拒绝来剪枝搜索空间。第二，可交换编码——用一元 NEG 和 INV 运算符替换减法和除法——将非交换二元运算缩减为仅有 POW，从而简化同构定义，同时保留标准符号回归函数集的全部表达能力。第三，标签感知剪枝回溯规范化算法能够高效地计算规范字符串；其可扩展性结果将在后续修订中报告。

在 Nguyen 和 AI Feynman 基准测试套件上进行的实证验证，确认了该表示的五项基本性质。所有测试表达式的往返保真度均达到 100%；剪枝算法与完整规范化算法在超过 99.97% 的测试用例中结果一致，同时生成的字符串长度最多比贪心基线短 21%。在 961 个有向无环图上，规范不变性和幂等性的验证结果均为 100%，这为猜想 2.11 提供了支持，也为剪枝规范字符串是完备的带标签有向无环图不变量提供了有力的实证依据。在每个测试点上，求值保持的误差均在容差 10^{-8} 以内。搜索空间分析证实，规范化会将所有 $\Theta(k!)$ 等价的节点编号表示归并为单一规范形式。规范字符串上的莱文斯坦距离在表达式同构类空间上构成有意义的度量：单字符编辑对应于运算符替换等语义上局部的变化，距离值也与表达式对之间的结构差异相吻合。对一个有代表性的表达式进行邻域分析，发现总体冗余率为 71.2%，量化了规范化带来的搜索空间压缩效果。

局限性。 关于两个核心猜想——往返保真度（猜想 2.10）和规范不变性（猜想 2.11）——的形式化证明仍未完成，将留待未来工作。关于剪枝规范化算法的 CPU 时间如何随有向无环图大小变化的可扩展性研究目前仍在分析中，并将在后续修订版中报告。与具体符号回归算法（如 GraphDSR 或 BINGO）开展集成实验，以适应度评估次数和解的质量为指标展示端到端搜索空间缩减，也同样留待未来工作。最后，当前字母表固定为表 1 中的函数集；将 ISALSR 扩展到用户定义运算符集、条件表达式或多输出有向无环图仍需进一步研究。

未来研究方向。 当前的优先事项包括对两个猜想进行形式化证明、完成全面的可扩展性研究，以及与最先进的符号回归求解器开展集成实验。长期研究方向包括在基于种群的方法中利用规范字符串作为去重键——以此替代对同构个体进行适应度重新评估——并将其用作基于语言模型的符号回归方法的结构化表示；在这种表示中，字母表提供了自然的词元词汇表，并内置有效性保证。莱文斯坦距离所诱导的度量结构还可能支持遗传编程中的保持多样性算子，从而显式地遍历规范字符串空间。

致谢

作者谨此感谢马拉加大学 SCBI（超级计算与生物信息学）中心提供的计算资源（Picasso 超级计算机）、技术专长和协助。

参考文献

John R. Koza。《遗传编程：通过自然选择进行计算机编程》。麻省理工学院出版社，马萨诸塞州剑桥，1992年。ISBN 978-0-262-11170-6。

Silviu-Marian Udrescu 和 Max Tegmark。AI Feynman：一种受物理学启发的符号回归方法。*ScienceAdvances*，6(16):eaay2631，2020年。doi：10.1126/sciadv.aay2631。NASA技术隶属机构。William La Cava、Patryk Orzechowski、Bogdan Burlacu、Fabrício Olivetti de França、Marco Virgolin、Ying Jin、Michael Kommenda 和 Jason H. Moore。现代符号回归方法及其相对性能。载于遗传与进化计算会议（*GECCO*）论文集，第1946–1954页，美国马萨诸塞州波士顿，2022年。计算机协会。doi：10.1145/3512290.3528718。NASA技术隶属机构。William La Cava、Patryk Orzechowski、Bogdan Burlacu、Fabrício Olivetti de França、Marco Virgolin、Ying Jin、Michael Kommenda 和 Jason H. Moore。现代符号回归方法及其相对性能。载于神经信息处理系统大会（*NeurIPS*）数据集与基准赛道，2021年。https://arxiv.org/abs/2107.14351。Ezequiel Lopez-Rubio。IsalGraph：面向图的指令集与语言。arXiv:2512.10429v2，2025年。https://arxiv.org/abs/2512.10429。Nguyen Quang Uy、Nguyen Xuan Hoai、Michael O'Neill、Robert I. McKay 和 Edgar Galvan-Lopez。遗传编程中的语义交叉：在实值符号回归中的应用。遗传编程与可进化机器，12(2):91–119，2011年。doi：10.1007/s10710-010-9121-2。Brenden K. Petersen、Mikel Larma、Trent McConaghy、Claudio Santiago、Sergio Lanza 和 Ruben Glatt。深度符号回归：通过风险偏好策略梯度从数据中恢复数学表达式。载于 国际学习表征会议（*ICLR*），2021年。https://openreview.net/forum?id=m5Qsh0kBQG。Pengwei Liu、Linge Li、Hao Wang、Zheng Dang 和 Bin Hu。利用图神经网络探索数学表达式以进行符号回归。*Neural Networks*，187:107405，2025年。doi：10.1016/j.neunet.2025.107405。Yanjie Xiang、Qing Liu、Shen Gao 和 Yunfeng Liu。GraphSR：采用交换算子编码的基于图的符号回归。载于 神经信息处理系统大会（*NeurIPS*），2025年。即将出版。Paul Kahlmeyer、Johannes Hirth、Frank Loebe 和 Joachim Hereth Correia。用于符号回归的均匀深度优先搜索。载于 第三十三届国际人工智能联合会议（*IJCAI*）论文集，第4020–4028页，韩国济州，2024年。国际人工智能联合会议。doi：10.24963/ijcai.2024/445。Vladimir I. Levenshtein。能够纠正删除、插入和反转的二进制码。苏联物理学文献，10(8):707–710，1966年。