arXiv:2603.11039v1 [cs.CL] 11 Mar 2026

图表示指令集

预印本

Ezequiel López-Rubio*计算机语言与计算机科学系
马拉加大学 路易·巴斯德大道，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 路易·巴斯德大道，35号 29071 马拉加，
西班牙mpascual@uma.es

2026年3月12日

摘要

我们提出了 IsalGraph，这是一种将任意有限简单图的结构表示为由九字符指令字母表构成的紧凑字符串的方法。该编码由一个小型虚拟机执行，该虚拟机包含一个稀疏图、一个存储图节点引用的循环双向链表（CDLL）以及两个遍历指针。指令要么使指针在循环双向链表中移动，要么向图中插入节点或边。该设计的一个关键特性是，每个 由 Σ 构成的字符串都能解码为一个有效图，不会到达任何无效状态。贪心式 *GraphToString* 算法能够在关于节点数的多项式时间内，将任意连通图编码为字符串；穷举回溯变体则通过在所有起始节点和所有有效遍历顺序中选取字典序最小的最短字符串，生成一个 规范字符串 w_G^*。我们在五个真实世界图基准数据集（IAM Letter LOW/MED/HIGH、LINUX 和 AIDS）上评估了该表示方法，并表明 Isal Graph 字符串之间的莱文斯坦距离与图编辑距离（GED）具有很强的相关性。综合来看，这些特性使 IsalGraph 字符串成为一种紧凑、同构不变且兼容语言模型的图结构序列编码，可直接应用于图相似性搜索、图生成和图条件语言建模。

关键词 图表示 · 邻接矩阵 · 指令序列 · 深度学习 · 语言模型 · 结构模式

1 引言

图是科学家和工程师可用的表现力最强的数据结构之一。分子化合物、社交网络、知识库、蛋白质相互作用网络和电路拓扑，都可以建模为由边连接的节点集合 [Zhou 等人， 2020，Khoshraftar 和 An， 2024，Ju 等人， 2024]。当代计算图处理面临的一项核心挑战是表示：应如何对图的结构进行编码，使其形式能够支持高效计算、泛化和下游学习？

最主要的答案是邻接矩阵。给定一个具有 $N = |V|$ 个节点的图 $G = (V, E)$，其邻接矩阵 $M_G \in \{0,1\}^{N \times N}$ 记录了哪些节点对彼此相连。邻接矩阵是谱图

*通讯作者。ITIS Software。马拉加大学。弗朗西斯科·佩尼亚洛萨建筑师街 18 号，29010，马拉加，西班牙

理论、代数图算法，以及几乎所有现有的图深度学习方法 [Kipf 和 Welling，2017，Hamilton 等人，2017，Veli˘ckovi´c 等人， 2018]。然而，它的局限性非常明显：无论图稀疏性如何，它都占用 $O(N^2)$ 的空间。此外，它本质上是二维的，因此无法直接供序列模型使用，例如循环网络或 Transformer。最后但同样重要的是，它破坏了置换等变性，因为其含义取决于为节点指定的任意顺序。

一种可能的替代研究路线，是将图编码为序列，以便输入序列模型。在当前大语言模型时代，这尤其引人注目，因为这类模型已展现出处理、生成和推理序列数据的非凡能力 [Vaswani 等人，2017，Devlin 等人， 2019]。挑战在于设计一种序列编码，使其具备以下特性：(i)紧凑，对于稀疏图，使用远少于 $O(N^2)$ 个符号；(ii)可逆，以便能够从字符串中精确恢复原始图结构；(iii)保留结构，使相似的图产生相似的字符串；以及 (iv)可规范化，即每个同构类都有唯一的代表字符串。

本文提出了**IsalGraph**，一种满足上述四项要求的新型序列图表示方法。该编码由一个小型虚拟机定义，虚拟机包含一个稀疏图、一个由图节点构成的循环双向链表（CDLL），以及两个遍历指针。九条指令可使指针沿 CDLL 移动，或向图中插入节点和边。执行指令语言中的任意字符串，均可将其解码为一张图；反过来，贪心算法可以将任意连通图编码为字符串。需要强调的是，定义字母表上的所有字符串都是有效的，即它们都能解码为一张图。规范字符串通过在所有起始节点和所有有效遍历顺序中最小化字符串长度而得到，从而构成具有形式化正确性保证的完备图不变量。我们之前的工作[López-Rubio， 2025] 与 IsalGraph 有很大不同，因为较早的方法要求节点具有固定顺序，且不使用由节点构成的循环双向链表。

本文结构如下。第2节介绍图表示方法。随后，第3节报告一项探索性计算实验的结果。最后，第5 节讨论结论。

2 方法

本节介绍 ISALGRAPH 的形式化机制。小节 2.1定义解释器状态和指令集。小节 2.1.3 描述字符串到图（S2G）算法。小节 2.2 介绍*GraphToString* （G2S）算法。小节 2.3 陈述规范字符串不变性猜想。小节 2.4 确立 ISALGRAPH 字符串度量与图编辑距离之间的拓扑关系。

2.1 指令集与字符串执行

2.1.1 解释器状态

ISALGRAPH 解释器在执行指令字符串时会同时维护三个组成部分。**定义 2.1**（解释器状态）。
ISALGRAPH 解释器状态是一个三元组 $\mathcal{S} = (G, \mathcal{L}, \pi)$，其中：

- $G = (V_G, E_G)$ 是一个逐步构建的有限简单图，其节点由连续的非负整数 $\{0, 1, \ldots, |V_G| - 1\}$ 标识。
- $\mathcal{L}$ 是一个基于数组的循环双向链表 (CDLL)，其节点以整数载荷的形式携带图节点索引。我们用 $\text{val}_{\mathcal{L}}(\ell)$ 表示 CDLL 节点 ℓ的载荷，用 next(ℓ)、prev(ℓ) 表示其在循环顺序中的后继节点和前驱节点。CDLL 索引空间与图节点索引空间彼此不同：CDLL 节点 ℓ并非图节点 $\text{val}_{\mathcal{L}}(\ell)$ 本身。
- $\pi = (\pi_1, \pi_2)$ 是一对指针，其中每个指针都是一个 CDLL 节点索引。 π_1 称为主指针， π_2 称为次指针。

初始状态。 执行任何指令之前，解释器处于以下初始状态 *te*:

(i) G 恰好包含一个节点（节点 0），且不包含任何边。
(ii) $\mathcal{L}$ 恰好包含一个节点，其载荷为图节点 0。(iii) 两个指针 π_1 和 π_2 均指向这个唯一的 CDLL 节点。

2.1.2 指令字母表

ISALGRAPH 字符串由九字符字母表构成

$$\Sigma = \{N, n, P, p, V, v, C, c, W\}.$$

每条指令的语义均在表1 中定义，并在下文进一步阐述。

表1：ISALGRAPH指令集。val$_{\mathcal{L}}(\ell)$表示存储为CDLL节点 ℓ载荷的图节点索引。 N/n (P/p)指令沿CDLL向前（向后）循环遍历。 V/v 指令始终创建从指针节点到 新节点的边；指针本身不 移动。对于有向图，指令 C 和 c 仅有的区别是。

指令	Type	对状态的影响(G， $\mathcal{L}$， π_1， π_2)
N	主级移动（向前）	$\pi_1 \leftarrow$下一页$_{\mathcal{L}}(\pi_1)$
P	主级移动（向后）	$\pi_1 \leftarrow$prev$_{\mathcal{L}}(\pi_1)$
n	次级移动（向前）	$\pi_2 \leftarrow$下一个$_{\mathcal{L}}(\pi_2)$
p	次级移动（向后）	$\pi_2 \leftarrow$前驱$_{\mathcal{L}}(\pi_2)$
V	通过主级进行节点插入	将新节点 u 添加到 G；将边 (val$_{\mathcal{L}}(\pi_1)$ u) 添加到 G；在 π_1之后立即将 u插入 $\mathcal{L}$ 。
v	通过次级进行节点插入	将新节点 u 添加到 G；将边 (val$_{\mathcal{L}}(\pi_2)$ u) 添加到 G；将 u插入到 $\mathcal{L}$ 中，紧接在 π_2之后。
C	边插入（主级 $\rightarrow$ 次级）	将边（值$_{\mathcal{L}}(\pi_1)$，值$_{\mathcal{L}}(\pi_2)$）添加到 G。对于无向图，还会添加反向边。
c	边插入（次级 $\rightarrow$ 主 mary)	向 G添加边(val$_{\mathcal{L}}(\pi_2)$, val$_{\mathcal{L}}(\pi_1)$)。等效于 C 对于无向图。
W	空操作	状态保持不变。

关键语义说明。 在 V和 v 指令中， u 的新CDLL节点被插入在指针当前CDLL节点之后，但指针本身并不会前进到该新节点。这意味着在执行 V 指令之后， π_1 仍然引用与执行该指令之前相同的CDLL节点。

每个字符串都是有效的。 ISALGRAPH字母表的一项关键设计特性是，每个字符串 $w \in \Sigma^*$ 都会解码为一个有效的有限简单图。任何指令都不会产生未定义或不一致的状态：指针移动会绕循环CDLL回绕，节点插入指令和边插入指令始终具有明确且确定的效果。

2.1.3 StringToGraph算法

S2G算法（算法1）从初始状态开始逐条执行ISALGRAPH 字符串指令，并返回结果图。

注 2.2。 对于无向图，add_edge(u, v)会将(u, v)和(v, u)同时插入邻接结构，因此指令 C 与 c 效果相同。对于有向图，它们因边方向而不同。

例2.3（解码VvNV）。 我们跟踪S2G(VvNV，有向 $= false$)：

(1) 初始化： $G = (\{0\}, \varnothing)$；循环双向链表 $= [0]$； $\pi_1 = \pi_2 = \ell_0$ （载荷0）。(2) V：添加节点1；添加边(0， 1)；循环双向链表 $= [0, 1]$； π_1仍位于 ℓ_0。(3) v：添加节点2；添加边(0， 2)；循环双向链表 $= [0, 1, 2]$（插入在 $\pi_2 = \ell_0$之后，因此在循环顺序中位于0之后、1之前——实际上为 $[0, 2, 1]$）； π_2仍位于 ℓ_0。(4) N： $\pi_1 \leftarrow$ next(ℓ_0)（当前循环顺序 $[0, 2, 1]$中的节点2）。(5) V：添加节点3；添加边(2， 3)；循环双向链表 $= [0, 2, 3, 1]$。

2.2 图到字符串转换

G2S 算法是 S2G 的逆过程：给定一个图 G 和一个起始节点 $v_0 \in V(G)$，它会生成一个 ISALGRAPH 字符串 w ，使得 S2G(w) $\cong G$。该算法采用贪心搜索，每一步都会寻找能执行有用结构操作的最小指针位移（以发出的指针移动指令数量计）。

算法1 S2G(w，有向)：字符串到图

需要： 字符串 $w \in \Sigma^*$；布尔值有向
确保： 图 G，使得 S2G(w) $= G$

初始化解释器状态
$G \leftarrow$ 新建一个包含单个节点 $u_0 = 0$ 的图，有向 $=$ 有向
$\mathcal{L} \leftarrow$ 新建循环双向链表；$\ell_0 \leftarrow \mathcal{L}$.插入_在.....之后($\varnothing$, u_0)
$\pi_1 \leftarrow \ell_0$；$\pi_2 \leftarrow \ell_0$
依次执行每条指令
对于每个字符 σ 在 w **执行**
    如果 $\sigma = N$ **那么**
        $\pi_1 \leftarrow \mathcal{L}$.下一个($\pi_1$)
    否则如果 $\sigma = P$ **则**
        $\pi_1 \leftarrow \mathcal{L}$.上一个($\pi_1$)
    否则如果 $\sigma = n$ **则**
        $\pi_2 \leftarrow \mathcal{L}$.下一个($\pi_2$)
    否则如果 $\sigma = p$ **则**
        $\pi_2 \leftarrow \mathcal{L}$.前一个($\pi_2$)
    否则，如果 $\sigma = V$ **那么**
        $u \leftarrow G$.添加_节点()
        G.添加_边(值$_{\mathcal{L}}(\pi_1)$ u)
        $\mathcal{L}$.在_之后插入(π_1, u)　　指针 π_1 不移动
    否则如果 $\sigma = v$ **则**
        $u \leftarrow G$.添加_节点()
        G.添加_边(值$_{\mathcal{L}}(\pi_2)$ u)
        $\mathcal{L}$.在_之后插入(π_2, u)　　指针 π_2 不移动
    否则如果 $\sigma = C$ **则**
        G.添加_边(值$_{\mathcal{L}}(\pi_1)$, 值$_{\mathcal{L}}(\pi_2)$)
    否则如果 $\sigma = c$ **则**
        G.添加_边(值$_{\mathcal{L}}(\pi_2)$, 值$_{\mathcal{L}}(\pi_1)$)
    否则如果 $\sigma = W$ **则**
        skip　　▷ no-op
    结束如果
结束循环
返回 G

2.2.1 配对生成与成本排序

每一步的搜索空间都是整数位移对 $(a,\ b) \in \{-M, \ldots, M\}^2$的集合，其中 M是当前节点数，a和 b 分别是主级指针和次级指针的移动步数（正值 = 表示向前，负值 = 表示向后）。一对位移的代价是其指针移动次数总和 $|a| + |b|$，等于将发出的$N/P/n/p$指令数量。

定义 2.4（排序后的位移对）。对于正整数 M，令

$$\mathcal{P}(M) \;=\; \Big\{(a,b) \;\Big|\; a,b \in [-M, M]\Big\}$$

按（$|a| + |b|$、$|a|$、a、b）按字典序递增排列。主级排序键 $|a| + |b|$ 使总指针移动次数最小化；次级键用于确定性地打破平局。

2.2.2 算法描述

该算法（算法 2）维护一个输出图 G_{out} 以及两个节点索引映射：ι（输入到输出）和 ι^{-1}（输出到输入）。之所以需要这些映射，是因为 G_{out} 是以自身的节点编号逐步构建的，而该编号可能不同于输入图的节点编号。

在每次迭代中，算法按顺序枚举各对 $(a, b) \in \mathcal{P}(M)$，并按优先级顺序尝试四种操作：

V（**通过主级节点**）暂定的主位置 $\tilde{\pi}_1$ 对应于输入图中的一个节点，该节点有一个未映射邻居。此时会插入一个新节点。

v **（通过次级节点）**同上，但使用暂定的次级位置 $\tilde{\pi}_2$。

C **（边，主级 → 次级）**这些暂定位置对应于输入图中两个彼此相邻的节点，但其对应的输出图节点尚未连接。

c **（边，次级 →主级）**与 C 相同，但顺序相反；仅对有向图有意义 .

对于第一个适用任一操作的节点对（a， b），将其提交：输出指针移动指令（$|a|$ 个 N 或 P的副本； $|b|$ 个 n 或 p的副本），附加结构指令（V、 v、 C或 c），更新实际指针，然后继续循环。当 G 中的所有节点和边均已插入后，算法终止。

算法 2G2S(G， v_0)：GraphToString（贪心）

要求： 连通图 $G = (V,\ E)$；起始节点 $v_0 \in V$
确保： 字符串 $w \in \Sigma^*$，其 S2G(w) 为 $\cong G$

▷ 初始化状态
验证所有节点均可从 v_0到达
$G_{\text{out}} \leftarrow$空图； $u_0 \leftarrow G_{\text{out}}$.添加_节点()
$\mathcal{L} \leftarrow$ 新建循环双向链表； $\ell_0 \leftarrow \mathcal{L}$.在_之后插入($\varnothing$, u_0)
$\pi_1 \leftarrow \ell_0$; $\pi_2 \leftarrow \ell_0$
$\iota \leftarrow \{v_0 \mapsto u_0\}$； $\iota^{-1} \leftarrow \{u_0 \mapsto v_0\}$
$n_{\text{left}} \leftarrow |V| - 1$; $e_{\text{left}} \leftarrow |E|$； $w \leftarrow \varepsilon$
▷ 主循环：继续，直到所有节点和边都已插入
当 $n_{\text{left}} > 0$ **或** $e_{\text{left}} > 0$**时**
    $M \leftarrow |V(G_{\text{out}})|$
    对于 (a, b)，**遍历** $\mathcal{P}(M)$
        $\tilde{\ell}_1 \leftarrow$遍历($\mathcal{L}$, π_1, a)； $\tilde{v}_1 \leftarrow \iota^{-1}[$值$_{\mathcal{L}}(\ \tilde{\ell}_1)]$
        $\tilde{\ell}_2 \leftarrow$walk($\mathcal{L}$, π_2, b)； $\tilde{v}_2 \leftarrow \iota^{-1}[$值$_{\mathcal{L}}(\ \tilde{\ell}_2)]$
        如果 $n_{\text{left}} > 0$ **且** $\exists\, c \in N_G(\tilde{v}_1)$ 满足 $c \notin \text{dom}(\iota)$ **则** ▷ V：通过主级的节点
            $u \leftarrow G_{\text{out}}$.添加_节点()； $\iota[c] \leftarrow u$； $\iota^{-1}[u] \leftarrow c$
            G_{out}.添加_边(val$_{\mathcal{L}}(\ \tilde{\ell}_1)$ u)； $\mathcal{L}$.在.....之后插入_($\ \tilde{\ell}_1$, u)
            $w \mathrel{+}=$ moves(a,主级) + V； $\pi_1 \leftarrow\ \tilde{\ell}_1$
            $n_{\text{left}} \mathrel{-}= 1$； $e_{\text{left}} \mathrel{-}= 1$; **中断**
        否则若 $n_{\text{left}} > 0$ **且** $\exists\, c \in N_G(\tilde{v}_2)$ 满足 $c \notin \text{dom}(\iota)$ **则** ▷ v: 次级节点
            $u \leftarrow G_{\text{out}}$.添加_节点()； $\iota[c] \leftarrow u$； $\iota^{-1}[u] \leftarrow c$
            G_{out}.添加_边(值$_{\mathcal{L}}(\ \tilde{\ell}_2)$ u)； $\mathcal{L}$.插入_之后($\ \tilde{\ell}_2$, u)
            $w \mathrel{+}=$ 移动(b,次级) + v； $\pi_2 \leftarrow\ \tilde{\ell}_2$
            $n_{\text{left}} \mathrel{-}= 1$； $e_{\text{left}} \mathrel{-}= 1$; **跳出**
        否则如果 $(\tilde{v}_2, \tilde{v}_1) \in E$ **并且** (值$_{\mathcal{L}}(\ \tilde{\ell}_2)$, val$_{\mathcal{L}}(\ \tilde{\ell}_1)$) $\notin E(G_{\text{out}})$**则** ▷ C
            G_{out}.添加_边(val$_{\mathcal{L}}(\ \tilde{\ell}_1)$, val$_{\mathcal{L}}(\ \tilde{\ell}_2)$)
            $w \mathrel{+}=$ 移动(a, pri) + 移动(b, sec) + C
            $\pi_1 \leftarrow \tilde{\ell}_1$； $\pi_2 \leftarrow \tilde{\ell}_2$； $e_{\text{left}} \mathrel{-}= 1$; **中断**
        否则如果 G 有向**并且** $(\tilde{v}_1,\ \tilde{v}_2) \in E$ **并且** (值$_{\mathcal{L}}(\ \tilde{\ell}_1)$, 值$_{\mathcal{L}}(\ \tilde{\ell}_2)$) $\notin E(G_{\text{out}})$**则** ▷ c
            G_{out}.添加_边(值$_{\mathcal{L}}(\ \tilde{\ell}_2)$, 值$_{\mathcal{L}}(\ \tilde{\ell}_1)$)
            $w \mathrel{+}=$ 移动(a,优先级) + 移动(b, 次优先级) + c
            $\pi_1 \leftarrow \tilde{\ell}_1$； $\pi_2 \leftarrow \tilde{\ell}_2$； $e_{\text{left}} \mathrel{-}= 1$; **跳出**
        结束 if
    结束 for
结束 while
返回 w

此处，walk($\mathcal{L}$， ℓ， a) 会返回沿着 $|a|$ 向前（如果 $a > 0$）或向后（如果 $a < 0$）步到达的 CDLL节点（从ℓ开始）。辅助函数 moves(a， 主级) 会输出 a 个 N 的副本（如果 $a \geq 0$）或 $|a|$ 个 P 的副本（如果 $a < 0$）；对于带有 n/p的次指针，也会以类似方式处理。

注释 2.5（可达性前置条件）。 对于有向图， V和 v 指令始终创建形如（已有_节点 → 新_节点）的边。因此，G2S 只能对可从 v_0 通过有向出边到达的节点进行编码。如果从所选起始节点无法到达任何节点，该算法将报错。

注释 2.6（字符串长度分解）。对于具有 G 个 N 节点和 M 条边的图，任何 ISALGRAPH 字符串编码的长度 G均满足：

$$|w| \;=\; \underbrace{(N-1)}_{\text{one } V/v \text{ per non-root node}} \;+\; \underbrace{M-(N-1)}_{\text{one } C/c/V/v \text{ per extra edge}} \;+\; \underbrace{\sum_{k}(|a_k|+|b_k|)}_{\text{pointer moves}}.$$

前两项由 G固定；只有指针移动总代价取决于遍历顺序。因此，最小化 $|w|$归结为最小化指针移动总量。

2.3 猜想性质

贪心G2S算法的基本形式并非标签无关：其邻居迭代顺序（针对集合）取决于节点从集合中提取的顺序，因此，节点编号不同的两个同构图可能会由贪心算法生成不同的字符串。为得到与标记无关的编码，我们通过穷举回溯定义规范字符串。

定义 2.7（规范字符串）。设 $\mathcal{W}(G)$表示由 G2S 的穷举回溯变体（在每个**分支点**探索所有有效邻居选择）从所有起始节点 $v \in V(G)$出发可生成的所有 ISALGRAPH 字符串的集合。 G 的规范字符串为

$$w_G^* \;=\; \operatorname{lexmin}\Big\{\, w \in \mathcal{W}(G) \;\Big|\; |w| = \min_{w' \in \mathcal{W}(G)} |w'| \Big\}.$$

也就是说，在 $\mathcal{W}(G)$ 的所有最短字符串中，按照 $C < N < P < V < W < c < n < p < v$ on Σ 上的全序，选出字典序最小的字符串。

这一构造启发了以下猜想，我们在此将其作为主要理论主张提出，并通过 t
在第.....节中通过实证研究得到 3.

猜想 2.8（规范字符串作为完备图不变量）。设 G 和 H 为有限简单图。那么

$$G \cong H \iff w_G^* = w_H^*.$$

正向结论（$G \cong H \Rightarrow w_G^* = w_H^*$）可由穷举规范搜索的标签无关性推出：同构 $\phi : V(G) \to V(H)$ 将从 v 出发遍历 G 的有效遍历集合与从 $\phi(v)$ 出发遍历 H 的有效遍历集合一一对应，因此两个图生成相同的字符串集合，进而具有相同的规范最小值。反向结论（$w_G^* = w_H^* \Rightarrow G \cong H$）可由往返正确性推出：如果 G 和 H 都生成相同的规范字符串 w，那么依据传递性， $G \cong \mathsf{S2G}(w) \cong H$ 。

要完成一份完整证明，必须严谨地证明穷举回溯算法确实是标签无关的，即其输出仅取决于输入的抽象邻接结构，而不取决于分配给节点的整数标识符。我们将这项验证留待未来研究，并在此提供实证支持：在九个图族中的 71 对同构与非同构图对上，不变性率和区分率均为 100%（见第3节）。注2.9（与图同构的关系）。如果猜想2.8 成立，那么计算 w_G^* 至少与图同构问题一样困难，因为 $w_G^* = w_H^*$ 当且仅当 $G \cong H$。已知图同构问题属于NP ，但尚不清楚它是否为NP完全问题；该问题已有准多项式时间算法。用于计算 w_G^* 的穷举规范搜索，其复杂度在最坏情况下会随 $|V(G)|$ 超多项式地增长。

2.4 拓扑结构

有用的图表示的一个关键属性是度量局部性：对图进行微小的结构变化，应当只导致其表示发生微小变化。相反，结构上不相似的图，其表示之间的距离应当很大。我们通过比较 ISALGRAPH 字符串上的莱文斯坦距离与标准的图编辑距离（图编辑距离，GED）来对此进行形式化。

2.4.1 字符串距离

定义 2.10（ISALGRAPH 字符串上的莱文斯坦距离）。对于两个 ISALGRAPH 字符串 $w_1,\ w_2 \in \Sigma^*$，其莱文斯坦距离为

$$d_{\mathrm{Lev}}(w_1, w_2) \;=\; \min\big\{k \;\big|\; w_1 \xrightarrow{k \text{ edits}} w_2\big\},$$

其中，单次编辑 是字符插入、删除或替换。通过标准动态规划，可在 $O(|w_1| \cdot |w_2|)$ 时间内完成计算。应用于规范字符串后，我们定义 ISALGRAPH图距离：

$$d_{\textsc{IsalGraph}}(G, H) \;=\; d_{\mathrm{Lev}}(w_G^*,\, w_H^*).$$

2.4.2 图编辑距离

定义 2.11 （图编辑距离 [Sanfeliu和Fu, 2012]）。图编辑距离 *GED(*G, H）是将G 转换为与 H同构的图所需的基本编辑操作（节点插入、节点删除、边插入、边删除）的最少数量，其中采用统一单位代价。

GED 是同构意义下有限图空间上的完备度量（即 GED(G, H) $= 0 \iff G \cong H$），但即使对于简单的单位代价函数，计算 GED 仍是 NP难问题。

2.4.3 局部性

我们陈述局部性 r 图表示与图编辑距离之间关系的主张。设 G 和 H 为有限、简单且连通的 d 图。记作 k = GED(G, H)，表示它们的图编辑距离。那么，根据经验：

(i) **单调性**。 $d_{\text{ISAL图表示}}(G, H)$是关于 k的非递减功能：向 G 添加更多编辑操作，会使规范字符串与 w_G^* 相距更远。(ii) **强相关性**。 在涵盖广泛图族的样本中，$d_{\text{ISAL图表示}}$ 与图编辑距离之间的皮尔逊相关和斯皮尔曼秩相关均较高。(iii) **敏感性**。 映射$k \mapsto d_{\text{ISAL图表示}}$ 平均而言单调递增。

局部性使 ISALGRAPH 有别于许多其他图表示。作为比较，（经过置换的）邻接矩阵之间的汉明距离不满足局部性，因为插入单个节点会改变矩阵中的 $O(N)$个条目。相较之下，ISALGRAPH 编码反映了重新编码修改后图所需的指令级代价，该代价自然受限于变化的边数加上到达新位置所需的额外指针移动。

对于 $N \times N$ 二元邻接矩阵，单次边插入恰好改变一个条目（对于无向图则改变两个），因此每次边编辑对应的汉明距离为 1 或 2——渐近地小于 ISALGRAPH 的界限。然而，若不先固定规范的节点排序，邻接矩阵就无法定义有意义的字符串度量，而这又会重新引入同构问题。ISALGRAPH 距离 $d_{\text{ISALGRAPH}}$ 在构造上具有同构不变性，而邻接矩阵上的汉明距离则不具备这一性质。

2.4.4 对图相似性搜索的启示

局部性具有实际意义。首先，这表明 $d_{\text{ISALGRAPH}}$ 可作为图编辑距离的计算高效的近似指标，用于相似性搜索：计算 d_{Lev} 需要 $O(|w_1| \cdot |w_2|)$ 时间，而精确图编辑距离则需要指数时间。其次，这种相关性足够强，使得 $d_{\text{ISALGRAPH}}$ 生成的排序与图编辑距离的排序高度一致，而这正是进行 k-近邻检索所需的性质。第三，由于每个 ISALGRAPH 字符串都能解码为有效图，因此在字符串空间中进行插值（例如通过随机或引导式编辑路径）会生成有效的中间图，从而能够通过字符串空间随机游走实现无梯度图优化。

3 计算实验

本节介绍实验所依据的数据集、评估方案和计算基础设施。实验设计围绕三个目标展开：(i) 量化真实世界图基准数据集中 ISALGRAPH 字符串上的莱文斯坦距离与图编辑距离之间的一致性；(ii) 刻画编码算法在合成随机图上的实证时间复杂度；以及 (iii) 衡量三种开销递增的编码策略之间编码质量与计算成本的权衡。

3.1 基准数据集

实验需要两类基准数据：带有精确图编辑距离真实值的真实世界图集合，用于评估莱文斯坦距离对结构差异度的近似准确性；以及规模受控的合成随机图，用于测量编码时间如何随节点数量变化。

3.1.1 真实世界图集合

来自三个应用领域的五个数据集用于相关性分析和加速比测量。在所有情况下，节点和边属性均被丢弃；ISALGRAPH 编码仅作用于图拓扑结构。由于G2S算法（算法 2）要求输入图是连通的，因此仅保留连通图。

IAM Letter（LOW、MED、HIGH）。IAM Letter 数据集 [Riesen 和 Bunke， 2008] 包含罗马字母表中 15 个大写字母的原型图，这些字母完全由直线组成。节点表示字母笔画的特征点（端点、拐角、交叉点），边连接同一笔画上的连续点。三个子集对应于施加在节点坐标上的不同程度的位置噪声：LOW、MED 和 HIGH。经过连通性过滤后，三个子集分别包含 1,180、1,253 和 2,059 个图，平均边数分别为 3.07、3.17 和 4.56。对于这三个子集，使用 NetworkX 的精确图编辑距离由 A* 算法计算 [Hagberg 等人提出的 2008]，采用统一单位代价：节点插入和删除的代价为 1，边插入和删除的代价为 1，节点替换的代价为 0（去除坐标后，所有节点在结构上均相同）。

LINUX。 LINUX 数据集包含从 Linux 内核子程序中提取的程序流图，最初由 Bai 等人 [2019] 收集，并由 Jain 等人 [2024]通过同构去重后重新发布。筛选出连通图并限制为最多含 12 个节点后，剩余 89 个图（平均边数：8.35）。直接使用 GraphEdX 仓库中预先计算的精确图编辑距离矩阵；这些矩阵通过 A* 搜索得到，仅采用拓扑结构成本（所有节点操作成本均为零；边插入和删除的成本均为单位成本）。

AIDS。 AIDS 数据集包含美国国家癌症研究所发展治疗学计划中的分子图，其中节点表示原子，边表示共价键。这里采用 Jain 等人 [2024] 发布的纯拓扑结构版本，其中所有节点标签（原子类型）均已剥除，并使用与 LINUX 相同的纯拓扑结构成本函数计算图编辑距离。筛选后，剩余 769 个连通图（平均边数：10.70）。

这五个数据集涵盖了从稀疏（平均边数为 3.07）到中等密度（10.70）的结构密度，样本规模则从 89 个图到 2,059 个图不等。表 2 汇总了每个数据集的图数量、有效成对比较数量以及平均边数。

3.1.2 合成图族

复杂度表征需要使用大小受控且独立于任何特定应用领域的图。随机连通图由两个标准图族生成：

- 巴拉巴西–阿尔伯特（BA）优先连接图 [巴拉巴西和阿尔伯特， 1999] 具有连接参数$m \in \{1, 2\}$。
- 厄尔˝什–Rényi（ER）随机图 [厄尔˝什和 Rényi， 1959] 具有边概率 $p \in \{0.3, 0.5\}$。当 ER 图不连通时，仅保留其最大连通分量。

对于每个图族，分别以节点数 $n \in \{3, 4, \ldots, 50\}$ 生成图以供贪心方法使用，以及以节点数 $n \in \{3, 4, \ldots, 20\}$ 生成图以供规范方法使用，每个实例的超时时间为 600 秒。每种$(n,$ 图族$)$ 组合生成五个独立实例。这些合成图仅用于图2所报告的时间复杂度分析；它们不参与相关性实验。

3.2 评估方案

评估分为四个部分：比较三种编码方法（第3.2.1节）；分析真实世界数据集上莱文施泰因距离与图编辑距离之间的相关性（第3.2.3节）；分别在合成数据集和真实世界数据集上测量复杂度和加速比（第3.2.4节）；以及针对一个小型示例图进行定性邻域分析（第3.2.5节）。

3.2.1 对比评估的编码方法

对G2S编码（第2.2节）中的三种变体进行评估，并按计算成本从高到低排列：

规范。 定义2.7中的穷举回溯过程，返回字典序最小的最短字符串 w^*_G。

贪心最小。 从每个起始节点 $v_0 \in V(G)$执行贪心G2S算法；选取所有运行结果中最短的字符串。

Greedy-rnd(v_0)。 从均匀随机选取的起始节点执行一次贪心 G2S 运行。

3.2.2 距离计算

精确图编辑距离（GED；定义2.11）作为结构差异性的真实值度量。每个数据集的图编辑距离计算过程见第3.1节。

对于每种编码方法，均根据所得的 ISALGRAPH 字符串计算全对莱文斯坦距离矩阵（定义2.10）。

3.2.3 相关性分析

在所有有效的上三角配对（i，j）上，量化莱文斯坦距离矩阵与图编辑距离矩阵之间的一致性，其中 $i < j$、图编辑距离（G_i，G_j）> 0，且 d_{Lev}（w_i，w_j）> 0。每个数据集和编码方法报告两项统计量：

- Spearman 秩相关系数 ρ，用于衡量两种距离度量之间的单调关联。
- 莱文斯坦距离对图编辑距离的普通最小二乘法（OLS）回归斜率 β，其中 $\beta = 1$ 表示标度相同，而 $\beta < 1$ 表示莱文斯坦距离的增长速度低于图编辑距离。

表 p 中报告的 p 值2 来自 SciPy 对斯皮尔曼检验的实现，该实现采用渐近 t 分布近似 $t = \rho\sqrt{(n-2)/(1-\rho^2)}$，自由度为 $n-2$，其中 n 表示有效配对数。鉴于五个数据集均有 $n > 1$，600 个配对，渐近近似是充分合理的。统计显著性按 $\alpha = 0.001$ 进行评估。

3.2.4 复杂度与加速比测量

编码时间在第3.1 节所述的合成图族上进行测量。每个图实例的每种编码重复 25 次，并取 CPU 时间中位数，以降低系统调度方差的影响。每个节点数 n 对应的汇总时间为各实例和图族的中位数，并以四分位距作为离散度量。通过对数变换数据（α 上拟合 $T(n)$，$= c \cdot n^{\alpha}$ 使用普通最小二乘法进行估算（$\log T$ 对 $\log n$），并以 R^2 报告拟合优度。
在五个真实世界数据集上测量 ISALGRAPH 流水线（编码加成对莱文斯坦距离）相较于精确图编辑距离的计算加速比。针对每对图计算加速比，并按图规模（$n = 3$至 11 个节点）分层，以几何平均数进行汇总。

3.2.5 邻域拓扑

为定性说明局部性（第 2.4 节），我们考察一个具有代表性的小图在图编辑距离和莱文斯坦距离下的邻域结构。基图为包含 5 个节点和 6 条边的房屋图。枚举与基图的图编辑距离为 GED $= 1$（单边编辑）的所有图，并计算它们与基编码之间的莱文斯坦距离。反过来，生成与基编码的莱文斯坦距离为 1 的所有字符串（单字符替换、插入和删除），通过 S2G 解码，并计算它们与基图之间的图编辑距离。

3.3 实现

IsalGraph 核心以 Python 实现，无需外部依赖。NetworkX [Hagberg 等人，2008], igraph [Csárdi 和 Nepusz，2006], 以及 PyTorch Geometric [Fey 和 Lenssen，2019]提供了与标准图形库的互操作能力。计时测量使用 `time.process_time()` 记录 CPU 时间，不包括 I/O 和系统调度所耗时间。所有实验均在马拉加大学超级计算与生物信息学中心（SCBI）的 Picasso 超级计算机上执行。为确保实验结果可复现，始终使用固定随机种子 42 。

4 结果

我们从三个方面评估 IsalGraph：莱文斯坦距离与图编辑距离（GED）的一致性（第 4.1 节），以及编码算法的实证时间复杂度（第 4.2 节）。此外，还通过定性邻域分析完成评估（第 4.3 节）。

4.1 与图编辑距离的相关性

表2 报告了每个数据集和编码方法中图编辑距离与莱文斯坦距离之间的斯皮尔曼等级相关系数 ρ。所有十五个 ρ数值均在统计上具有显著性 $\alpha = 0.001$。

在包含稀疏图（$\bar{m} \leq 4.56$）的三个 IAM Letter 子集中，规范编码与图编辑距离达成了较强的单调一致性：$\rho = 0.934$ LOW 上为 0。876，MED 上为 0。682，高于 HIGH。贪心最小仅以小幅差距落后于规范编码（分别为 $\Delta\rho = 0.027$、0。014 和 0。057），而 Greedy-rnd（v_0）的损失更大，在 LOW 上低于规范编码$\Delta\rho = 0.228$。

在较密集的 LINUX 和 AIDS 数据集（$\bar{m} = 8.35$ 和 10.70）上，相关性显著下降。LINUX 上表现最佳的方法是 Greedy-min（$\rho = 0.445$）；这是唯一一个该方法优于规范编码的数据集（$\rho = 0.433$），优势为 0.012。AIDS 上则由规范编码领先（$\rho = 0.349$）。LINUX 上的微小差异可能反映了有效配对数量（$n = 1{,}685$）相较于其他数据集（$n \geq 131{,}148$）较少，因此个别异常值配对对秩相关统计量的影响会被放大。

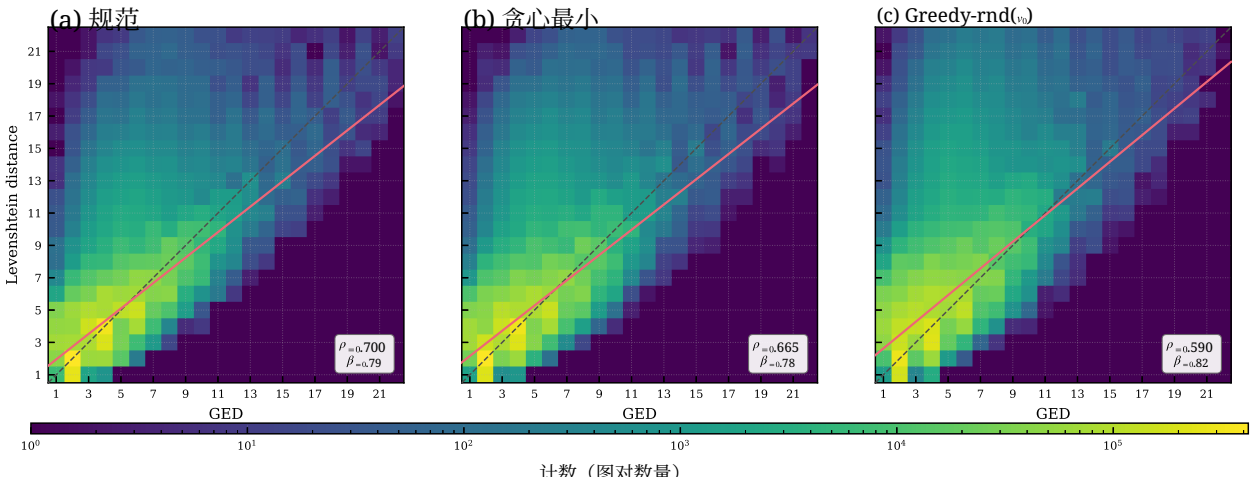


图1：五个基准数据集上图编辑距离（GED）与 Levenshtein距离之间的汇总相关性。整数坐标 (i, j) 处的每个单元格表示 GED 为 $= i$ 且 Lev 为 $= j$ 的图对数量（对数尺度；浅色表示 = 图对较少，深色表示 = 图对较多）；白色单元格表示没有观测到图对。灰色虚线：恒等线（Lev $=$ GED）。红色实线：普通最小二乘法（OLS）回归线。（a）规范编码（$n = 3,424,764$ 对，$\rho = 0.700, \beta = 0.79$）。（b）贪心最小编码（$n = 3,424,764$ 对，$\rho = 0.665, \beta = 0.78$）。（c）Greedy-rnd（$v_0$）编码（$n = 3{,}424,764$ 对，$\rho = 0.590, \beta = 0.82$）。报告的统计量：$\rho$ 表示 Spearman 秩相关系数，用于衡量两种距离度量之间的单调关联。β 表示 OLS 回归斜率；$\beta = 1$表示莱文斯坦距离与 GED 的尺度相同，而 $\beta < 1$则表示 Levenshtein距离的增长速度慢于 GED。

表2：数据集属性，以及不同编码方法下图编辑距离与 IsalGraph 莱文斯坦距离之间的斯皮尔曼 ρ 相关性。$\bar{m}$：每个图的平均边数（复杂度代理指标）。展示了每个数据集最佳方法之间的斯皮尔曼-ρ 差异。每个数据集的最佳 ρ 方法以**粗体**显示。

		IAM LOW	IAM MED	IAM HIGH	LINUX	AIDS
Prop.	N	1,180	1,253	2,059	89	769
	配对	695,610	784,378	2,118,711	1,685	131,148
	$\bar{m}$	3.07	3.17	4.56	8.35	10.70
Spear. ρ	规范	**0.934*****	**0.876*****	**0.682*****	0.433*** (-0.012)	**0.349*****
	贪心最小	0.908*** (-0.027)	0.862*** (-0.014)	0.625*** (-0.057)	**0.445*****	0.304*** (-0.045)
	Greedy-rnd(v_0)	0.706*** (-0.228)	0.682*** (-0.195)	0.577*** (-0.105)	0.301*** (-0.144)	0.251*** (-0.098)

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$。随着 ρ 在各数据集中逐渐退化，$\bar{m}$ 单调递增。

图1 展示了五个数据集中所有3,424,764 个有效配对中，图编辑距离与莱文斯坦距离的聚合联合分布。质量集中在恒等线附近，证实这两种度量总体上呈同单调关系，不过在较高的图编辑距离值处，分布范围有所扩大。OLS回归斜率 $\beta = 0.79$ (规范),0.78 (贪心最小)，以及0.82 (Greedy-rnd)始终低于 1，表明莱文斯坦距离的增长速度低于图编辑距离。这种压缩源于有界指令字母表（$|\Sigma| = 9$）：结构上相距较远的图，其编码字符串仍可能共享较长的公共子序列，从而减弱测得的字符串不相似度。

在所有五个数据集中，图密度与相关强度之间都呈现明显的单调关系。平均边数 $\bar{m}$ 从3.07 (IAM Low) 增至 10.70 (AIDS) 时， ρ 对每种编码方法都呈下降趋势（表2）。降幅最大的是 IAM HIGH（$\bar{m} = 4.56$，规范 $\rho = 0.682$）与 LINUX（$\bar{m} = 8.35$，规范 $\rho = 0.433$）之间：平均边数近乎翻倍，同时 ρ 相对下降了 37%。这种性能下降与 G2S 遍历的序列性质相符：随着边密度增大，单次深度优先遍历所捕获的图的成对连通性比例逐渐降低，所得字符串也就成为对完整拓扑结构更粗略的近似。

4.2 实证时间复杂度

图 2显示了三种方法在合成随机图（巴拉巴西–阿尔伯特和 Erd n ős–Rényi 系列）上的编码时间中位数随图规模的变化˝。对数变换数据上的幂律拟合 $T(n) = c \cdot n^{\alpha}$ 得到的指数为 $\alpha = 3.1$ ，对应 Greedy-rnd(v_0)、 $\alpha = 4.5$ Greedy-min，以及 $\alpha = 9.0$ Canonical，所有拟合的 $R^2 \geq 0.979$。

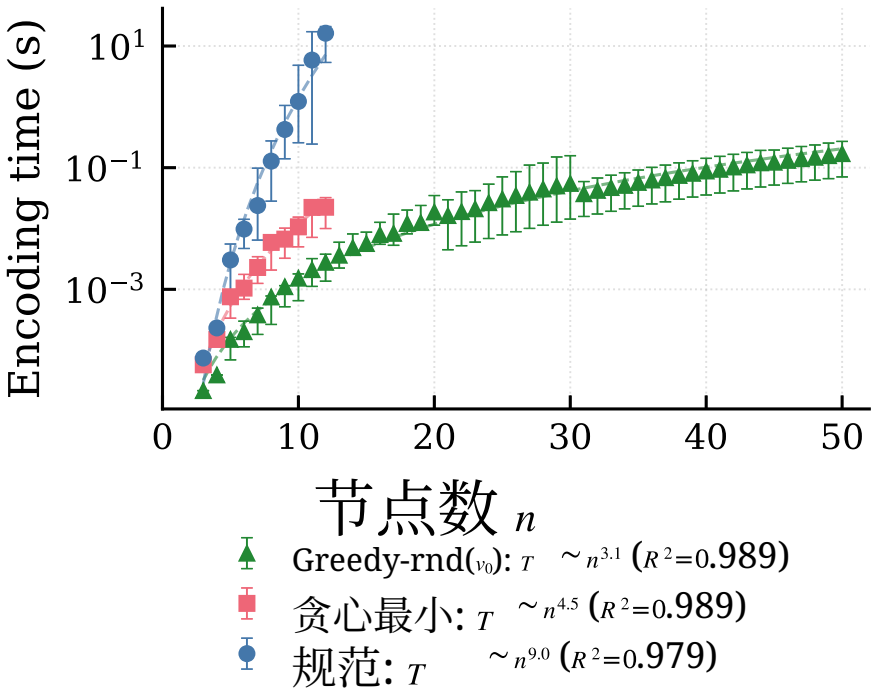


图2：ISALGRAPH 编码方法在随机图（巴拉巴西–阿尔伯特 $m \in \{1, 2\}$和 Erd˝os–Rényi $p \in \{0.3, 0.5\}$）上的实证时间复杂度。横轴：节点数 n；纵轴：编码时间（秒，对数尺度）。标记表示各图实例的中位数；误差线表示四分位距。虚线为通过普通最小二乘法（OLS）对双对数数据进行拟合得到的多项式拟合 $T = c \cdot n^{\alpha}$ 。Greedy-rnd(v_0)： $\alpha = 3.1$, $R^2 = 0.989$。Greedy-Min： $\alpha = 4.5$,$R^2 = 0.989$。Canonical： $\alpha = 9.0$, $R^2 = 0.979$。贪心方法呈现多项式级增长（$\alpha \approx$ 3–5），而规范方法在随机图上的增长为超多项式（$\alpha \approx 9$），并且当规模超过 $n \approx 12$ 后变得不可行。

Greedy-rnd 的指数 $\alpha \approx 3$ 与单次 G2S 遍历的成本一致，其主要操作是在每个已访问的 $O(n)$ 个节点上重复执行邻居选择步骤。Greedy-min 对所有 n 个起始节点迭代执行贪心过程，使经验指数升至 $\alpha \approx 4.5$；高于 n^4 的半个单位反映了不同起始节点对应的字符串长度各不相同，以及相关的比较成本。在 600 秒超时限制内，这两种贪心变体均可处理包含 50 个节点的图。

规范方法表现出 $\alpha = 9.0$，这是对所有起始节点和所有有效邻居排列进行穷举回溯的直接结果。在 $n = 12$时，规范编码已接近超时阈值；超过这一规模后，若不进一步改进算法，实际应用将不切实际。较高的 R^2 值证实，在测试范围内，幂律模型能够捕捉到观察到的标度律；不过，由于遍历顺序的组合爆炸，规范方法的真实渐近复杂度是超多项式的。

4.3 邻域结构

图 3 通过一个具体示例展示了图空间与字符串空间邻近性之间的关系，补充了上述总体相关性分析。基图 G_0 是房屋图（5 个节点、6 条边），其规范 ISALGRAPH 编码用作参考字符串。

G_0 的图编辑距离为1的邻居由10个非同构图组成，这些图通过一次保留连通性的边插入或删除获得（6次删除和4次插入）。它们与 G_0 的编码之间的莱文斯坦距离范围为1至5：一次结构编辑可能需要在指令字符串中进行最多五次字符变化。这种差异源于规范编码选择全局最优遍历顺序；修改一条边可能使这一最优顺序完全改变，从而生成显著不同的字符串，尽管底层图仅发生了最小变化。

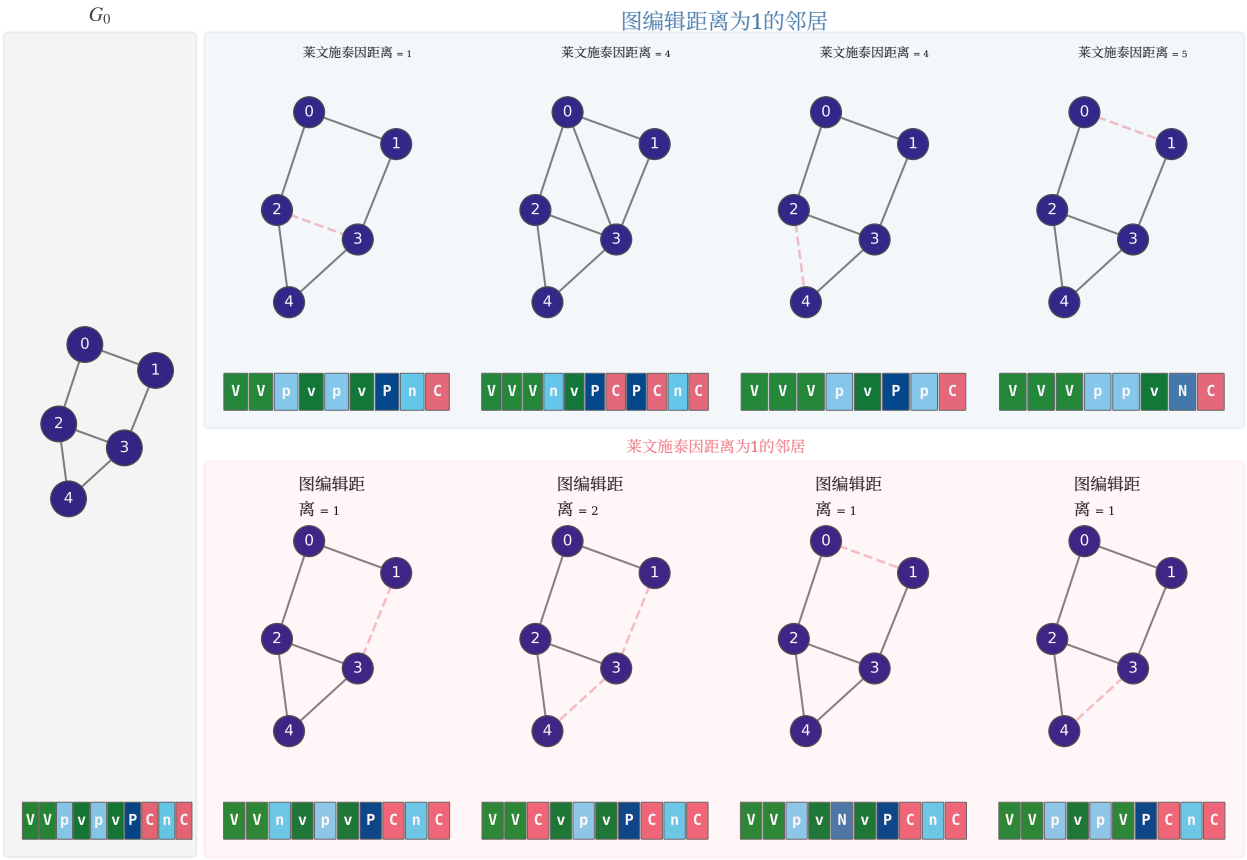


图3：房屋图 G_0 （5个节点、6条边）在两种距离度量下的邻域拓扑。中间列：带有其规范 IsalGraph 编码的基图 G_0（按指令类型进行颜色编码）。顶部各行：4个具有代表性的图编辑距离为1的邻居（单次边编辑），其与G_0的编码之间的莱文施泰因距离为 Lev $\in [1, 5]$ 。底部各行：4个具有代表性的1-莱文斯坦邻居（对指令字符串进行单次字符替换、插入或删除），其图编辑距离为 GED $\in [1, 2]$。红色虚线边表示与 G_0之间的结构差异。每个图下方的水平热图以逐字符着色的方式呈现 IsalGraph 指令字符串（字母表$\Sigma = \{N, n, P, p, V, v, C, c, W\}$）。图编辑距离为1的邻域与1-莱文斯坦邻域之间的不对称性表明，图空间中的接近并不意味着字符串空间中的接近，反之亦然。

反过来，1-莱文施泰因邻域——即与基编码仅有一次替换、插入或删除差异的字符串——对应的图，其图编辑距离为 $\in \{1, 2\}$ 至 G_0。因此，字符串空间中的邻近意味着图空间中的邻近：对指令字符串进行微小扰动，只会造成微小的结构变化。这种方向上的紧密关系源于指令语义：每个字符至多对应一种结构操作（节点创建、边插入或指针移动），从而限制了单个字符变化对拓扑结构的影响。

两个邻域之间的不对称性——从字符串空间到图空间的映射紧密，从图空间到字符串空间的映射宽松——是任何允许多个遍历顺序表示同一张图的编码所固有的。这带来一个实际影响：ISALGRAPH 字符串上的莱文斯坦距离更可能高估图编辑距离（当

较小的结构变化需要大幅字符串重排时），而不是低估它（因为每次字符变化所产生的结构影响都有上限）。这种保守偏差有利于召回率而非精确率：基于莱文斯坦距离的检索更可能返回一张略有差异的图，而不是漏掉一张确实相似的图；在优先考虑召回率的检索场景中，这一特性很有优势。

5 结论

贡献总结。 本文介绍了 ISALGRAPH，这是一种用于有限简单图的顺序指令表示。该编码由一台包含九条指令的虚拟机定义，虚拟机通过两个遍历指针操作一个存储图节点引用的循环双向链表（CDLL）；执行指令时，节点和边会被插入稀疏图中。由此得到的表示具有四项特性，使其有别于现有图编码：

(i) **普遍有效性。** 字母表 $\Sigma = \{N, n, P, p, V, v, C, c, W\}$ 上的每个字符串都能解码为一个有效的有限简单图。不存在语法或语义上无效的字符串，因此无需设计用于检查有效性的解码器，也简化了生成模型的设计。(ii) **可逆性。** 贪心 GraphToString（G2S）算法可将任意连通图 G 编码为字符串 w，使得 S2G(w) $\cong G$。在涵盖十二个图族的 945 个测试实例中，往返正确性的通过率经确认达到 100%；此外，还通过 VF2同构算法进行了独立交叉验证。(iii) **典范完备性（推测）。** 通过对所有起始节点和所有有效邻居排列进行穷举回溯计算得到的规范字符串 w^*_G，据推测是一个完备图不变量：$G \cong H \iff w^*_G = w^*_H$。这一推测得到九个结构族中 71 个图对测试结果的支持，涵盖树、圈、完全图、星图、轮图、巴拉巴西–阿尔伯特图和彼得森图等；测试中的不变性和区分准确率均为 100%。(iv) **度量局部性。** 在真实世界图基准测试中，ISALGRAPH 字符串之间的莱文斯坦距离与图编辑距离呈强相关；在稀疏的 IAM Letter (LOW) 数据集上，斯皮尔曼 $\rho = 0.934$（$n = 695$，610 个图对，$p < 0.001$），且在测试的全部五个数据集中均保持显著相关；这些数据集覆盖了从 $\bar{m} = 3.07$ 到每个图的平均边数为 $\bar{m} = 10.70$ 的不同结构密度。

局限性。 有三项局限性值得明确指出。首先，典范完备性猜想尚未得到证明。要给出形式化证明，必须严格证明穷举回溯算法是标签无关的，即其输出仅取决于抽象邻接结构，而不取决于分配给节点的整数标识符。其次，规范编码的规模呈超多项式增长（$T \sim n^{9.0}$），对于超过约 12 个节点的图，在当前硬件上，若超时时间设为 600 秒，则计算上不可行。第三，G2S 算法要求输入图是连通的；对于有向图，还要求所有节点都能通过有向出边从选定的起始节点到达。不满足这些条件的图无法在未经预处理的情况下进行编码。

致谢

作者谨此致谢。 感谢计算资源（Picasso超级计算机）、技术专长和协助。
由 SCBI 提供（ 马拉加大学超级计算与生物信息学中心。

参考资料

Jie Zhou、Ganqu Cui、Shengding Hu、Zhengyan Zhang、Cheng Yang、Zhiyuan Liu、Lifeng Wang、Changcheng Li 和 Maosong Sun。图神经网络：方法与应用综述。*AI Open*，1:57–81，2020。doi:10.1016/j.aiopen.2021.01.001。Shima Khoshraftar 和 Aijun An。图表示学习方法综述。*ACM*智能系统与技术汇刊，15(2):1–45，2024。doi:10.1145/3633518。Weiyu Ju 等。深度图表示学习综合综述。神经网络，171:1063–1095，2024。doi:10.1016/j.neunet.2024.106207。Thomas N. Kipf 和 Max Welling。基于图卷积网络的半监督分类。载于国际学习表征会议，2017。arXiv:1609.02907。William L. Hamilton、Rex Ying 和 Jure Leskovec。大型图上的归纳式表示学习。载于神经信息处理系统进展，第30卷，第1024–1034页，2017。

Petar Veli˘ckovi´c、Guillem Cucurull、Arantxa Casanova、Adriana Romero、Pietro Liò 和 Yoshua Bengio。图注意力网络。载于国际学习表征会议，2018。arXiv:1710.10903。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力机制就是你所需要的一切。载于神经信息处理系统进展，第30卷，2017。Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。BERT：面向语言理解的深度双向变换器预训练。*arXiv*预印本，2019。arXiv:1810.04805。Ezequiel López-Rubio。用指令序列表示图结构。*arXiv*预印本，2025年。arXiv:2512.10429v2。Alberto Sanfeliu 和 King-Sun Fu。用于模式识别的属性关系图距离度量。*IEEE*系统、人和控制论汇刊，(3):353–362，2012。Kaspar Riesen 和 Horst Bunke。用于基于图的模式识别与机器学习的IAM图数据库仓库。载于结构、句法与统计模式识别，第5342卷，《计算机科学讲义》，第287–297页。施普林格，2008。doi:10.1007/978-3-540-89689-0_33。Aric A. Hagberg、Daniel A. Schult 和 Pieter J. Swart。使用 NetworkX 探索网络结构、动力学和功能。载于第七届科学计算中的*Python*会议论文集（*SciPy 2008*），第11–15页，2008。Yunsheng Bai、Hao Ding、Song Bian、Ting Chen、Yizhou Sun 和 Wei Wang。SimGNN：一种快速计算图相似度的神经网络方法。载于第十二届*ACM*国际网络搜索与数据挖掘会议论文集，第384–392页。ACM，2019。doi:10.1145/3289600.3290967。Eeshaan Jain、Indradyumna Roy、Saswat Meher、Soumen Chakrabarti 和 Abir De。使用神经集合散度计算具有一般代价的图编辑距离。载于神经信息处理系统进展，第37卷，2024。arXiv:2409.17687。Albert-László Barabási 和 Réka Albert。随机网络中标度律的出现。《科学》，286(5439):509–512，1999。doi:10.1126/science.286.5439.509。Paul Erd˝os 和 Alfréd Rényi。关于随机图 I。《德布勒森数学出版物》，6:290–297，1959。Gábor Csárdi 和 Tamás Nepusz。用于复杂网络研究的 igraph 软件包。《*InterJournal*复杂系统》，1695:1–9，2006。Matthias Fey 和 Jan Eric Lenssen。使用 PyTorch Geometric 进行快速图表示学习。载于*ICLR*图与流形表示学习研讨会，2019。arXiv:1903.02428。