arXiv:2512.15157v1 [cs.DB] 17 Dec 2025

# 提取节点比较洞见，以交互方式探索属性图

Cristina Aguiar
*ICMC*
圣保罗大学
圣卡洛斯，巴西
0000-0002-7618-1405

Jacques Chabin
*LIFO UR4022*
奥尔良大学，*INSA CVL*
奥尔良，法国
0000-0003-1460-9979

Alexandre Chanson
*LIFAT*
图尔大学
图尔，法国
0000-0001-9195-5950

Mirian Halfeld-Ferrari
*LIFO UR4022*
奥尔良大学，*INSA CVL*
奥尔良，法国
0000-0003-2601-3224

Nicolas Hiot
*LIFO UR4022*
奥尔良大学，*INSA CVL*
奥尔良，法国
0000-0003-4318-4906

Nicolas Labroche
*LIFAT*
图尔大学
图尔，法国
0000-0002-2794-2124

Patrick Marcel
*LIFO UR4022*
奥尔良大学´，*INSA CVL*
奥尔´良，法国
0000-0003-3171-1174

Ver´onika Peralta
*LIFAT*
图尔大学
图尔，法国
0000-0002-9236-9088

Felipe Vasconcelos圣保罗大学圣卡洛斯数学与计算机科学研究所˜保S˜圣卡洛斯，巴西
0009-0007-7922-9746

**摘要——几十年来，研究人员一直通过节点评分来了解图中节点的重要性（例如中心性），但据我们所知，尚未有人基于属性图中节点的属性对节点进行比较。本文提出一种方法，用于自动提取属性图中节点的比较结果，以支持对这些图进行交互式探索性分析。我们首先介绍一种利用待比较节点的上下文来设计比较指标的方法。随后，我们对如何利用这些指标对节点进行分组这一问题作出形式化定义，使提取的比较既有意义，又非显而易见。我们提出了多种启发式方法来解决这一问题。在真实属性图数据库上的测试表明，简单的启发式方法可在数分钟内获得洞见，而要获得更高质量的洞见，则需要使用速度较慢的启发式方法。**

索引词—**探索性数据分析、属性图、比较**

## I. 引言

设想一位分析师通过交互方式探索属性图，以发现有意义的模式和解释。本文探讨如何在属性图中自动生成节点比较洞察。我们将节点比较洞察定义为一对在上下文上相似的节点，其节点属性、边属性或图拓扑所派生的一组数值指标存在显著差异。这些指标有助于分析师聚焦于信息量最大的对比，从而让探索过程更有针对性、效率更高。据我们所知，这是首项正式定义并研究属性图中比较洞见生成问题的工作。

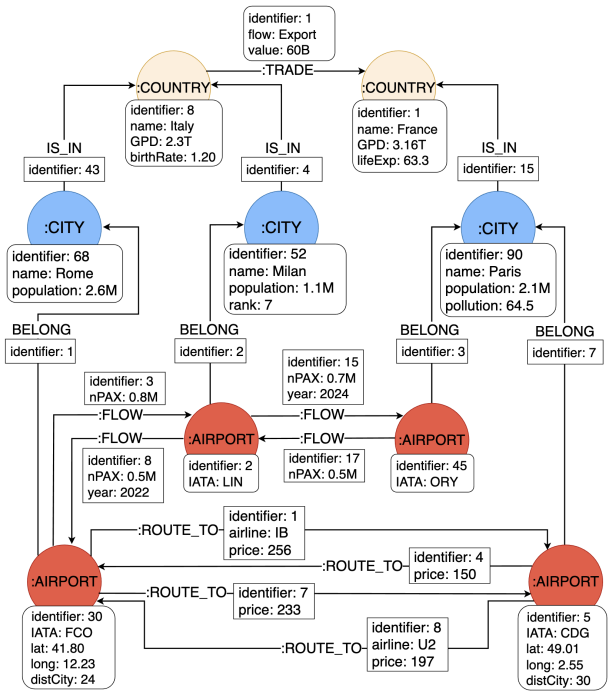



图 1：示例 1 的属性图 $G$。

**示例 1**。请看图 *1* 中关于国家间航空路线和贸易的属性图。该图展示了我们测试数据库中的一部分数据，其中节点代表实体，边对应实体之间的关系。位于顶部的浅橙色节点表示国家，特征为

其特征是具有国内生产总值和出生率等数值属性。位于中间的蓝色节点代表城市，其数值属性包括人口（以百万为单位）和污染指数。显示在底部的红色节点代表机场，其数值属性包括纬度和经度，以及国际航空运输协会代码等标识符。

边刻画了不同类型的关系。关系 *FLOW* 表示两个机场节点之间的旅客交换，并包含属性“旅客人数”，用于指定乘客数量。关系 *ROUTE_TO* 记录机场之间各种形式的运输，无论运输的是货物还是人员。

图 *1* 中的示例包含两个国家（意大利和法国）、三个城市（罗马、米兰和巴黎）以及四个机场（*FCO*、*LIN*、*CDG* 和 *ORY*）。

假设我们希望从该图中提取机场之间有意义的比较。为此，需要定义一组指标，以捕捉可用于评估机场的、可比较且有意义的方面。这些指标既可以依据图的结构属性，也可以依据相连实体的属性。例如，可以考虑机场与相邻机场之间不同航线的数量，即统计与该机场相连的不同国际航空运输协会代码（$i_1$）。另一个相关指标（$i_2$）是通往相邻机场的航线平均价格，因为关系 *ROUTE_TO* 包含用于指定连接价格的属性。还可以根据节点在图中的位置相关信息派生出其他指标，例如机场所在城市的人口（$i_3$），或其所属国家的国内生产总值（$i_4$）。这些指标为机场之间的多维比较提供了依据。

表 *I* 报告了上述四项指标在整个数据库上计算所得结果的一个示例，而非仅针对图 *1* 所示的抽取部分。

□

| 机场 | 指标 | | | |
|---|---|---|---|---|
| | $i_1$ | $i_2$€ | $i_3$(M) | $i_4$(T) |
| CDG | 285 | 525 | 2.1 | 3.16 |
| FCO | 242 | 450 | 2.6 | 2.30 |
| LIN | 51 | 425 | 1.1 | 2.30 |
| ORY | 178 | 400 | 2.1 | 3.16 |

表 I：示例 1 数据集的指标示例

指标计算完成后，为了找出有趣的比较，分析师可能会先对数据进行归一化，只保留相关指标中的一个，将相似的机场分组，然后比较每组中的机场。在这个例子中，FCO、CDG 和 ORY 的人口相近，因此可以归为一组，而 LIN 则归入另一组。在 FCO、CDG 和 ORY 之间，分析师会指出，尽管这些机场属于同一组，但它们在航线和乘客数量方面存在差异。

自动找出这些比较会引出以下问题：

- 应使用哪些比较指标?
- 如何从图中提取这些指标?
- 如何将指标划分为用于分组和用于比较的两类?

我们提出以下贡献来回答这些问题：

- 一种基于图中节点上下文设计指标的方法。我们通过节点类型之间的关系以及在图实例中检测到的这些关系的基数，对此上下文进行形式化定义，
- 对属性图中用于比较节点的指标进行的初步分类，
- 选择用于比较节点的指标以及用于对节点进行分组的指标这一问题的形式化定义。尽管该问题类似于聚类，但其目标有所不同，即簇应最大化簇内比较的显著性，
- 用于解决上述问题的各种启发式方法，
- 在真实图上进行的一系列测试，以评估所提出启发式方法的效率和有效性，
- 一个基于开放数据的基准数据集，其中的一个摘录用于上文的示例 1，以表示用于设计比较指标的各种属性（节点和边的属性）。

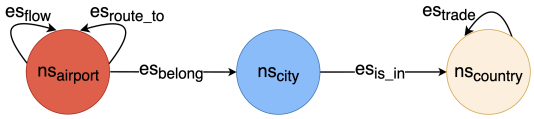
本文的结构如下。第 II 节介绍背景定义。第 III 节详细说明我们设计比较指标的方法，第 IV 节形式化阐述使用这些指标提取比较洞察的问题。第 V 节介绍用于解决这些问题的算法解决方案。第 VI 节介绍所开展的测试。第 VII 节讨论相关工作，第 VIII 节总结全文并展望未来。

## II. 背景

我们基于 [3], [4] 中的定义来定义属性图及其模式，并引入相对于关系端点的关系基数概念。

**定义 1**（属性图）。设 $\mathcal{O}$ 是一个标识符集合，$\mathcal{L}$ 是一个标签集合，$\mathcal{P}$ 是一个属性名称集合，$\mathcal{V}$ 是一个值集合。属性图定义为一个元组 $G = (N, E, \rho, \lambda, \nu)$，其中：

- $N \subset \mathcal{O}$ 是一个有限的节点标识符集合；
- $E \subset \mathcal{O}$ 是一个有限的边标识符集合，$N \cap E = \emptyset$
- $\rho$： $E \to N \times N$ 是一个将边映射到节点有序对（边的端点）的全函数。我们分别用函数 $\rho^{src}$ 和 $\rho^{tgt}$ 表示 $E$ 中一条边的源节点和目标节点；
- $\lambda$： $N \cup E \to \mathcal{L}$ 是一个为节点和边分配标签的全函数；
- $\nu : (N \cup E) \times \mathcal{P} \rightharpoonup \mathcal{V}$ 是一个用于定义节点和边的属性的部分函数。




| 类型名称 | 标签 | 属性集合 |
|---|---|---|
| $ns_{airport}$ | :机场 | 标识符、国际航空运输协会、纬度、经度、距城市距离 |
| $ns_{city}$ | :城市 | 标识符、名称、人口、污染、排名 |
| $ns_{country}$ | :国家 | 标识符、名称、GPD、出生率、预期寿命 |
| $es_{belong}$ | :属于 | 标识符 |
| $es_{is_{in}}$ | :位于 | 标识符 |
| $es_{flow}$ | :FLOW | 标识符，旅客人数，年份 |
| $es_{route_{to}}$ | :航线至 _ | 标识符，航空公司，价格 |
| $es_{trade}$ | :贸易 | 标识符，FLOW，值 |

图 2：示例 1 的属性图类型 $S$

$G$ 中的一条边定义了其连接的两个节点之间的关系。

为简化表示，我们假设对于所有属性图，具有相同标签的边，其源节点和目标节点的标签也分别相同。形式化地说，以下性质成立：$\forall e \in E, \forall e' \in E$ 若 $\lambda(e) = \lambda(e')$ 则$\lambda(\rho^{src}(e)) = \lambda(\rho^{src}(e'))$ 且 $\lambda(\rho^{tgt}(e)) = \lambda(\rho^{tgt}(e'))$□

根据定义 1，在作为贯穿示例的属性图中，连接机场与城市的边，以及连接城市与国家的边，必须具有不同的标签。

示例 *2*。设 $G = (N, E, \rho, \lambda, \nu)$ 为示例 *1* 中的图。考虑两个节点 $n_1$ 和 $n_2$，它们分别描述机场 *CDG* 和巴黎市；另有一条边 $e_1$ 表示 *CDG* 是属于巴黎的机场。即：$n_1, n_2 \in N$, $e_1 \in E$, $\rho(e_1) = (n_1, n_2)$, $\lambda(n_1) = Airport$, $\lambda(n_2) = City$,$\lambda(e_1) = BELONG$, $\nu(n_1, IATA) = CDG$, $\nu(n_2, name) = Paris$, $\nu(n_2, population) = 2.1M$。

我们还需要考虑属性图的模式（或类型）。为此，我们首先介绍形式基础类型的概念。形式基础类型 $T$是一个有序对 $(l, P)$，其中$l \in \mathcal{L}$ 且 $P \subseteq \mathcal{P}$。我们用 $\mathcal{T}$表示所有形式基础类型构成的集合。

**定义 2** (属性图类型)。属性图类型（下文简称图类型）是一个元组 $S = (N_S, E_S,$元素类型$s, \rho_S)$，其中：

- $N_S$ 和 $E_S$ 是互不相交的有限集合，分别包含节点类型名称和边类型名称；
- 元素类型$s$：$N_S \cup E_S \to \mathcal{T}$将元素（节点或边）类型名称映射到一种形式基础类型；
- $\rho_S$：$E_S \to N_S \times N_S$ 是一个全函数，将边映射到节点的有序对（即该边的端点）。我们分别用函数 $\rho_S^{src}$和 $\rho_S^{tgt}$ 表示边（属于 $E_S$）的源节点和目标节点（属于 $N_S$）。□

**示例 3.** 在图*2*中，我们展示了运行示例上下文中的图类型 $S_1$。它定义了三种节点类型，即 $N_S = \{ns_{airport}, ns_{city}, ns_{country}\}$，以及五种边类型，即 $E_S = \{es_{trade}, es_{is\ in}, es_{belong}, es_{flow}, es_{route\ to}\}$。我们来看看 $es_{belong}$ 及其关联的节点。

我们有：元素类型$s(ns_{airport})$=(:机场，{标识符, 国际航空运输协会,纬度,经度, 距城市距离}),元素类型$s(ns_{city})$=(:城市,{标识符, 名称, 人口, 污染, 排名}), 元素类型$s(es_{belong})$ = (:属于,{标识符}),$\rho_S(es_{belong}) = (ns_{airport}, ns_{city})$。□

**定义 3**（元素有效性与类型实例）。给定属性图 $G$，其中的元素 $x$（节点或边）$G$具有标签 $l_1$ 和属性名称集合 $P_1$，若其满足形式基础类型 $T = (l, P)$，则称其有效。我们也称这样的元素是 $T$ 的实例（记作 $x \in Inst(T)$）。□

**定义 4**（图类型的实例）。给定图类型 $S = (N_S, E_S,$元素类型$s, \rho_S,)$ 以及属性图 $G = (N, E, \rho, \lambda, \nu)$，若 $G$ 是 $S$ 的实例（记作 $G \in$*Inst*（$S$））， 则有：

- 对于每个节点 $n \in N$，都存在 $n_S \in N_S$，使得 $n$ 满足元素类型$s$（$n_S$），且
- 对于每条边 $e \in E$：– 存在 $e_S \in E_S$，使得 $e$ 满足元素类型$s$（$e_S$）；– $\rho^{src}$（$e$） 满足元素类型$s$（$\rho_S^{src}$（$e_S$））， 且– $\rho^{tgt}$（$e$） 满足元素类型$s$（$\rho_S^{tgt}$（$e_S$））。

我们也称 $G$相对于 $S$是有效的。□

**示例 4.** 图 *1* 中的图是图*2*所示图类型的一个实例。代表机场 *CDG* 和 *LIN* 的节点等，虽然具有不同的属性集，但都满足节点类型 $ns_{airp}$所定义的属性，因此在该实例中有效。□

接下来，我们通过考察给定图实例中的所有实例，来定义关系相对于其端点的基数。考虑图2中的图类型 $S$ 及图 1 中与之有效的实例 $G$。要确定 $es_{belong}$相对于 $S$中源节点 $ns_{airport}$ 的基数，我们针对 $G$中 $ns_{airport}$ 的每个实例 $n$，统计从 $n$出发的出向$BELONG$ 关系实例的数量。在 $G$中，每个机场恰好属于一个城市，因此 $card^{src} = 1$。同样，要确定 $es_{belong}$ 相对于 $S$中目标节点 $ns_{city}$ 的基数，我们针对$G$中 $ns_{city}$ 的每个实例 $n$，统计指向 $n$的入向 $BELONG$ 关系实例的数量。在 $G$中，由于巴黎有两个机场，因此 $card^{tgt} = *$。以下定义将形式化地阐述这一概念。

**定义5**（关系相对于其端点的基数）。设 $G = (N, E, \rho, \lambda, \nu)$ 是一个符合图类型 $S = (N_S, E_S,$ 元素类型$s, \rho_S)$ 的属性图。给定 $e_S \in E_S$,

- 如果 $\exists n \in$ *Inst(ElemTypes*($\rho_S^{src}$(*e*$_S$))) 使得$|\{e \in$ *Inst(ElemTypes*($e_S$)) $| \rho^{src}(e) = n\}| > 1$ 则 $card^{src}\ e_S = *$ $card^{src}\ e_S = 1$ () 否则 () 如果
- $\exists n \in$ *Inst(ElemTypes*($\rho_S^{tgt}$ ($e_S$))) 使得$|\{e \in$ *Inst(ElemTypes*($e_S$)) $| \rho^{tgt}(e) = n\}| > 1$ 则$card^{tgt}(e_S) = *$ 否则 $card^{tgt}(e_S) = 1$。□

现在介绍图上路径的表示法。

**定义 6 (路径)。** 给定一个图 $G = (N, E, \rho, \lambda, v)$，我们将路径定义为由节点和边组成的序列，其形式为$p = (n_1, \alpha_1, ..., \alpha_k, n_{k+1})$，其中

- *for* $k = 0$ *(empty path),* $p = (n_1)$, $\lambda(p) = \epsilon$;
- *for* $k > 0$,
  - $\forall i \in [1, k], n_i \in N$, $\alpha_i \in \{e_i, e_i^{-1}\}$, $e_i \in E$ *and* $\rho(e_i) = (n_i, n_{i+1})$;
  - $e^{-1}$ *is used to indicate that an edge* $e$ *is traversed in its reverse direction and*
  - $\lambda(p) = (\lambda(\alpha_1).....\lambda(\alpha_k))$.

为简便起见，节点 $n$、$n'$ 之间的路径 $p$ 记作 $n \xrightarrow{\lambda(p)} n'$。图类型上的路径也以类似方式定义。

当 $G$ 相对于图类型 $S$有效时，$G$ 中的每条路径 $n \xrightarrow{\lambda(p)} n'$ 都对应于$S$中的一条路径 $p_S = n_S \xrightarrow{\lambda(p)} n'_S$：对于 $p$ 中的每个节点 $n_i$ 和每条边 $e_i$（或 $e_i^{-1}$），都存在 $e_{i_S} \in E_S$，使得 $e_i \in \mathit{Inst}(e_{i_S})$、$n_i \in \mathit{Inst}(\rho^{src}(e_{i_S}))$ 且 $n_{i+1} \in \mathit{Inst}(\rho^{tgt}(e_{i_S}))$（若边反向遍历，则反之亦然）。此外，当 $\forall e_{i_S} \in p_S$满足 $card_S^{src}(e_{i_S}) = 1 \vee card_S^{tgt}(e_{i_S}^{-1}) = 1$时，我们记作 $n_S \xrightarrow[1]{\lambda(p)} n'_S$；否则记作 $n_S \xrightarrow[*]{\lambda(p)} n'_S$。 □

**示例 5。** 在图 *1* 中，我们将节点 $n_1$ 称为代表米兰的节点，将节点 $n_2$ 称为代表罗马的节点，而节点$n_3$ 和 $n_4$ 则分别对应它们各自的机场。路径 $p_1 = (n_1, BELONG^{-1}, n_3, FLOW, n_4, BELONG, n_2)$ 是 $G$ 中的一条路径，路径 $ns_{City} \xrightarrow[*]{es_{BELONG}^{-1}.es_{FLOW}.es_{BELONG}} ns_{City}$和 $ns_{Airport} \xrightarrow[1]{es_{BELONG}.es_{IS\ IN}} ns_{Country}$ 则是 $S$ 中的路径。 □

### III. 设计比较指标

据我们所知，尚未有人研究过如何设计用于比较属性图中节点的指标。我们借鉴了在知识图谱中识别比较特征的思路 [16]。这一思路首先是为每个待比较的节点识别一个比较上下文。然后，根据上下文中发现的属性以及上下文拓扑，设计候选指标。

#### *A.* 指标来源与计算

指标非常重要，因为它们能够将复杂的图节点表示为可比较且有意义的描述符，从而支持数据分析。通过同时编码语义内容和结构特征，它们确保比较既考虑数据本身，也考虑上下文，同时保持透明且可复现。然而，一项关键挑战在于如何定义恰当的指标——确定要捕捉数据和拓扑的哪些方面，以及如何对其进行度量。表 II 总结了计算指标的不同方法，我们将其区分为：

- 来源：指标值来自节点属性、图拓扑（例如入度），或两者兼有（例如加权度，即连接带有权重时边权重的求和），

| 描述 | 来源 | 作用域 | 计算 |
|---|---|---|---|
| $n$的属性 | 属性 | 内部 | 原始 |
| $l$ 或 $r \in p$的属性 | | | |
| 包含 $n \xrightarrow[1]{\lambda(p)} l$ | 属性 | 上下文 | 原始 |
| 与 $n \xrightarrow[*]{\lambda(p)} l$ | 属性 | 上下文 | 派生 |
| $n$ 与关系 $r$的度 | 拓扑 | 上下文 | 派生 |
| $n$的中心性 | 拓扑 | 全局 | 派生 |

表 II：节点 $n$的指标示例

- 作用域：指标值来自节点本身（内部）、源自节点上下文（即其邻居节点），或是全局的（即相对于完整图而言），
- 计算：如果指标值直接来自属性值，则为原始；如果指标值涉及计算（例如，对多个值进行聚合），则为派生。

例如，机场所在城市的人口是一个基于属性、上下文相关的原始指标，因为其值是从机场的上下文中获取的人口这一原始属性。相比之下，诸如到 PageRank 得分最高的节点的最短路径百分位数这样的指标，则是一个基于拓扑的全局派生指标。

本文重点关注从节点及其邻域的数值属性派生的指标。这里，邻域由节点的上下文定义。该上下文涵盖节点所属的层次结构及其最近邻节点。在示例 1 中，比较两个机场需要考虑以下因素：(i) 机场节点本身的属性，(ii) 机场上下文中节点的属性（即各机场所属的城市和国家），(iii) 相邻机场的属性，以及 (iv) 连接这些节点的关系的属性。下一节将对上下文的概念进行形式化定义。

#### *B.* 节点上下文

上下文的概念是在图类型$S$上正式定义的。为方便起见，我们有时会说节点 $n$在 $G$中的上下文，实际指的是图类型 $S$中对应节点 $n_S$ 的上下文。要确定图类型 $S$中节点 $n_S$ 的上下文，我们按以下步骤进行：

1) 考虑 $S$中与对应节点 $n_S$ 相连的所有关系。2) 考虑 $S$ 中 $n_S$ 的层次环境。具体而言，这意味着收集所有形式为 $p_S = n_S \xrightarrow[1]{\lambda(p)} n'_S$，的路径，并将 $p_S$ 上出现的每个元素添加到 $n$的上下文中。

形式上，节点的上下文是一个子图 $S_1 \subseteq S$。$G$ 中的一个实例是一个子图 $G_1 \subseteq G$，它相对于 $S_1$是有效的。

例如，在图2中，节点 $ns_{airport}$的上下文对应于图2中的整个图，但不包括边 $es_{trade}$。请注意，节点 $ns_{City}$ 和 $ns_{Country}$表示节点 $ns_{airport}$的层次环境。

下面的定义形式化地阐述了上下文的概念。

**定义 7 (**节点上下文**)**。令 $S$ $=(N_S, E_S,$元素类型$_S, \rho_S)$ 为一种图类型。 $n_S \in N_S$ 的上下文定义为 $S$的一个子图，记作$C_{n_S}=(N_{n_S}, E_{n_S},$ 元素类型$_{n_S}, \rho_{n_S})$，其中:

- $N_{n_S}=\{n_S\}\cup\{n'_S \in N_S | \exists p, n_S \xrightarrow[1]{\lambda(p)} n'_S\}\cup\{n'_S \in N_S | \exists e_S \in E_S, (n_S, n'_S)=\rho_S(e_S)\}$;
- $E_{n_S}=\{e_S \in E_S | \exists p, n_S \xrightarrow[1]{\lambda(p)} n'_S, e_S \in p\}\cup\{e_S \in E_S | (n_S, n'_S)=\rho_S(e_S)\}$;
- $\mathit{ElemType}_{n_S}=\mathit{ElemType}_{S|_{E_{n_S}\cup N_{n_S}}}$;
- $\rho_{n_S}=\rho_{S|_{E_{n_S}}}$ □

*C.* 候选指标

定义上下文后，我们将与该上下文中节点和边相关联或由其派生的所有数值属性（例如，邻居数量）视为候选指标。这些指标所用的值也取决于涉及关系的基数。为了一般性起见，我们不考虑全局范围的指标（例如中心性），因为这类指标可以预先计算并记录为节点属性，从而成为节点上下文的一部分。

对于关系 $e_S$，其基数$^{src}(e_S)=1$（或基数$^{tgt}(e_S)=1$ 在反向遍历边时），可以直接使用属性值，例如城市的人口或国家的国内生产总值。相反，对于基数$^{src}(e_S)=*$的关系（或反向遍历时基数$^{tgt}(e_S)=*$），则需要聚合相邻元素的属性，例如计算相邻机场的平均员工人数，或计算向这些机场发送或从其接收的平均乘客人数。

指标是支持数据图中进行有意义比较的数值属性。为了有效考察指标的值，通常有必要检查其轨迹——也就是说，需要了解这些值是在图中的什么位置获取的。为准确捕捉这些信息，我们将指标名称定义为四元组，用于描述所考察属性的来源，即获取该属性所遵循的路径、涉及的属性、该属性来自节点还是边，以及所涉及的聚合函数。

**定义 8 (**属性上的运算符**)**。设 $p=n_S \xrightarrow{\lambda(p)} n'_S$是上下文 $C_{n_S}$上的一条路径。设 $prop \in \mathcal{P}$ 是 $p$中最后一个节点或最后一条边的属性。设 $\mathcal{A}$ 是一个聚合函数集合（通常包括求和、平均值、最小值、最大值和计数），并设 $OpDict$：$\mathcal{P}\to 2^{\mathcal{A}}$ 是一个领域字典，用于指明每个属性最适用的聚合函数（例如，人口适用求和，污染适用平均值）。

我们将 $Op(p, prop)$ 定义为与路径中的属性 $ro$ 相关联的运算符集合，$p$具体如下: $p$ $fi$ $f$

$$Op(p,prop)=\begin{cases}\{\mathit{id}\} & \text{if } n_S \xrightarrow[1]{\lambda(p)} n'_S \text{ or } p=(n_S)\\ OpDict(prop) & \text{if } n_S \xrightarrow[*]{\lambda(p)} n'_S\end{cases}$$

其中 *id* 是恒等函数。 □

**定义 9 (**上下文的候选指标**)**。令 $C_{n_S}=(N_{n_S}, E_{n_S},$ 元素类型$_{n_S}$, $\rho_{n_S})$为节点类型 $n_S$ 的上下文。令 $p$ 为图 $C_{n_S}$ 上的一条路径。

指标名称是一个四元组 $\langle path, prop, type, op\rangle$，其中

- $path$对应路径 $p$的标签 $\lambda(p)$，或者在空路径时对应 $\epsilon$;
- $prop \in \mathcal{P}$ 是属性名称;
- $type \in$ {'节点'、'边'} 表示该属性来自节点类型，还是来自指向该节点的边类型;
- $op \in Op(p, prop)$ 是路径中该属性的运算符。

对于路径 $p=(n_S, e_{S1}, \ldots, e_{Sk}, n'_S)$，其中包含 $(L_n, P_n)$ =元素类型$_S(n'_S)$、$(L_e, P_e)$ = 元素类型$_S(e_{Sk})$，集合 $I^p_{n_S}$是以下各项的并集:

{$\langle\lambda(p)$ $prop$, "节点" $op\rangle | prop \in P_n$, $op \in Op(p, prop)$} ∪{$\langle\lambda(p)$ $prop$, "边" $op\rangle | prop \in P_e$, $op \in Op(p, prop)$}

节点类型 $n_S$ 的指标集 $I_{n_S}$ 是 $\bigcup_p I^p_{n_S}$，对于所有路径 $p=n_S \xrightarrow{\lambda(p)} n'_S$，其中 $C_{n_S}$。 □

**示例 6.** 让我们考虑图*2*中图类型里的上下文 $ns_{airport}$（第 *III-B* 节）。作为$ns_{airport}$的指标，我们列举如下:

- $i_1=\langle es_{route\ to}, IATA,$ '节点' $count\rangle$，表示相邻机场的总数。
- $i_2=\langle es_{route\ to}, price,$'边' $avg\rangle$，表示飞往相邻机场的航线平均价格。
- $i_3=\langle es_{belong}, population,$'节点' $id\rangle$，表示该机场所在城市的人口。
- $i_4=\langle es_{belong}.es_{is\ in}, GPD,$ '节点' $id\rangle$，表示该机场所在国家的 *GPD*。 □

**定义 10 (**指标值**)**。令 $G=(N, E, \rho, \lambda, \nu)$为关于图类型 $S=(N_S, E_S,$元素类型$_S, \rho_S)$的有效图。令 $n_S \in N_S$ 为一种节点类型，$I_{n_S}$ 为一个指标集。

节点 $n \in$*Inst*(元素类型$_S(n_S)$) 的指标值以及指标 $i=\langle path_i, prop_i, type_i, op_i\rangle \in I_{n_S}$ 定义如下。令 $N_i=\{n' \in N | \exists p=n \xrightarrow{path_i} n'\}$，即通过 $path_i$可到达的节点，并令 $E_i=\{e' \in E | \exists p=(n, \ldots, e', n') \wedge p=n \xrightarrow{path_i} n'\}$，即 $path_i$ 的最后边。

指标 $i$ 的值为:

$$\mathit{value}(n,i)=\begin{cases}null & \text{if } N_i=\emptyset\\ \mathrm{op}_i(\{\nu(n', prop_i) | n' \in N_i\}) & \text{if } type_i=\text{'node'}\\ \mathrm{op}_i(\{\nu(e', prop_i) | e' \in E_i\}) & otherwise\end{cases}$$

□

**示例 7.** 对于示例 *6* 中的指标，如果我们仅考虑图 *1* 所示的值，就会得到表 *III* 中的结果。请记住，图 *1* 仅展示了我们数据集中的一小部分，而表 *I* 则列出了整个测试数据集中的相同指标。 □

**定义 11** （指标值的集合。）。令 $S$ 为一种图类型， $S$ 为 $S$ 的一种节点类型。值

| 机场 | 节点 ID | 指标 | | | |
|---|---|---|---|---|---|
| | | $i_1$ | $i_2$ (C) | $i_3$ (M) | $i_4$ (T) |
| CDG | 30 | 2 | 173.5 | 2.1 | 3.16 |
| FCO | 28 | 0 | 244.5 | 2.6 | 2.3 |
| LIN | 5 | 0 | *null* | 1.1 | 2.3 |
| ORY | 45 | 2 | *null* | 2.1 | 3.16 |

表 III：根据图 1 中的值计算出的指标值。

节点类型 $n_S$ 的属性 $prop$ 的值为 $values(n_S, prop) = \{\nu(n, prop) | n \in Inst(ElemType_S(n_S))\}$。节点类型 $n_S$ 的指标 $i$ 的值，是所有 $n \in Inst(ElemType_S(n_S))$ 对应的 $value(n, i)$ 的并集，即 $value(n_S, i) = \bigcup_{Inst(ElemType_S(n_S))} value(n, i)$；而 $n_S$ 中节点类型的所有指标的值，是所有 $i \in I_{n_S}$ 对应的 $value(n_S, i)$ 的并集，即 $value(n_S) = \bigcup_{I_{n_S}} value(n_S, i)$。□

### D. 验证候选指标

设 $n_S$ 为一种节点类型，$i = \langle \lambda(p)\ prop, type, agg \rangle$ 为 $I_{n_S}$ 中的一个指标，其中 $value(n_S, i)$ 是值的集合 $\{value(n, i) | n \in Inst(ElemType(n_S))\}$。设 $x_{prop}$ 为定义 $prop$ 所针对的元素类型，即若 $p = \epsilon$，则 $n_S$ 是 $p$ 中按类型确定的最后一条边或最后一个节点，否则为其他情况。若 $i$ 满足以下理想的验证属性[1]（其中 $\alpha, \beta$ 为常数），则认为其相对于有效指标集合 $I$ 和属性名称集合 $P$ 有效：

- 可接受方差：$1 \le \alpha \le |value(n_S, i)| \le \beta \le |Inst(ElemType(n_S))|$
- 可接受密度：$|value(x_{prop}, prop)| \ge \gamma$
- 非冗余性：对于某个函数 $f$，$\forall i' \in I, value(n_S, i) \ne f(value(n_S, i'))$
- 上下文化：对于某个函数 $\forall v \in value(n_S, i)\ v = f(i, C_{n_S})$，$f$，
- 缩放：$value(n_S, i) \in [0, 1]$
- 已舍弃属性：对于属性名称的集合 $P$，$prop \notin P$

可接受方差允许舍弃值过少或过多的指标，因为它们通常被视为缺乏信息。可接受密度允许舍弃空值过多的指标。非冗余性允许舍弃相关指标，即其值可由其他指标推导出的指标。上下文化要求指标的值是其在上下文中所在位置的函数。这可用于降低通过从待比较节点出发的长路径找到的属性值的权重。缩放要求对指标进行归一化，以免偏向其中某一个。最后，已舍弃属性可视为用户定义的对不需要属性的排除（例如代理键或标识符、邮政编码、纬度/经度等）。这类验证属性常用于机器学习。

[1]我们使用“验证属性”这一术语，以区别于图属性；在本文中，后者简称为属性。

## IV. 问题定义

上一节详细说明了如何针对给定节点类型，通过收集并聚合节点上下文中的属性来定义一组指标。本节将探讨如何基于这组指标提取给定类型节点的比较。

### A. 节点比较洞察

**定义 12**（比较／分组方案）。令 $G = (N, E, \rho, \lambda, \nu)$ 是相对于图类型 $S = (N_S, E_S, 元组类型_S, \rho_S)$ 的有效图。令 $n_S \in N_S$ 为一种节点类型，$I_{n_S}$ 为 $n_S$ 的指标集合。节点比较方案（分别地，分组方案）是一个元组 $(n_S, I)$，其中 $I \subseteq I_{n_S}$ 是用于比较（分别地，分组）的指标集合。

给定一个比较方案 $(n_S, I)$，比较是一个元组 $(n, n', I)$，其中 $n, n' \in Inst(元素类型(n_S))$. 令 $c = (n, n', I)$ 为一个比较，且 $i$ 为 $I$ 中的一个指标。该比较的显著性 $c$ 对于 $i$ 由数学表达式 $sig_i$ 给出。例如，$sig_i(n, n') = |value(n, i) - value(n', i)|$。比较的总体显著性通过聚合 $I$ 的所有指标的显著性（例如使用求和）得到。

下文的目标是最大化比较的显著性。然而，为了排除那些显而易见的比较，我们要求比较在相似的节点之间进行，因此引入了分组方案。给定 $g = (n_S, I)$ 这一分组方案、$I$ 中的指标 $i$，以及 $n, n' \in Inst(ElemType(n_S))$，$i$ 的 $n$ 与 $n'$ 之间的距离由数学表达式 $dist_i$ 给出，例如 $dist_i = (value(n, i) - value(n', i))^2$。$g$ 的总体距离由聚合 $I$ 所有指标对应的距离得出。

定义 *13*（节点比较洞察）。设 $G = (N, E, \rho, \lambda, \nu)$ 是一个关于图类型 $S = (N_S, E_S, ElemType_S, \rho_S)$ 的有效图。设 $n_S \in N_S$ 是一种节点类型，且 $n, n' \in Inst(ElemType(n_S))$。给定比较方案 $c = (n_S, I)$ 和分组方案 $g = (n_S, I')$，其中 $I \cap I' = \emptyset$，$n$ 与 $n'$ 的节点比较洞察是元组 $(n, n', \{(value(n, i)\ value(n', i)) | i \in I\})$。

下文将提取比较洞见的问题形式化为寻找用于分组和比较节点的方案，以最大化比较的显著性，并最小化所比较节点之间的距离。

### B. 问题表述

给定节点类型 $n_S$ 的指标集 $I_{n_S}$，以及待比较的节点集合 $N' = Inst(ElemType(n_S))$，提取比较洞察的问题在于找到一个比较方案和一个分组方案。换言之，我们要在 $I_{n_S}$ 的所有指标中，寻找一个子集 $P_S$，使得 $(n_S, P_S)$ 是一个比较方案；并寻找一个与 $P_S$ 不相交的子集 $P_D$，使得 $(n_S, P_D)$ 是一个分组方案。我们不要求 $P_S \cup P_D = I_{n_S}$，也就是说，我们实际要寻找的是一种.....

3 划分 $I_{n_S}$，其中 $P_U = I_{n_S} \backslash (P_S \cup P_D)$ 是既不用于比较也不用于分组的指标集合。

直观地说，该问题的一般形式可以写为：

$$\text{argmax} \sum_{k=1}^{K} \Big( \sum_{n,n' \in S_k} \sum_{i \in P_S} sig_i(n,n') - \sum_{n,n' \in S_k} \sum_{i \in P_D} dist_i(n,n') \Big)$$

可以将其理解为：寻找 $P_S$ 和 $P_D$，以计算由节点组成的 $K$个簇 $\{S_1, ..., S_K\}$，同时最小化簇内距离（第二项），并最大化簇内的显著性（第一项）。

接下来，我们考虑问题的一个版本，其中使用差值的绝对值来表示比较的显著性，并使用欧几里得距离进行分组。此外，我们还对问题中的每一项加权，以平衡各指标在簇之间的分布，以及各指标在 $P_S$ 和 $P_D$之间的分布。

因此，我们要寻找 $P = \{P_S, P_D, P_U\}$的一个三划分 $I_{n_S}$，以定义由节点组成的 $K$ 个簇，从而得到以下优化问题：

$$\begin{aligned} \underset{\{S_1,\ldots,S_K\},P}{\text{argmax}} \quad & \sum_{k=1}^{K} \frac{1}{|S_k|} \frac{|P_S|}{|P|} \\ & \sum_{n,n' \in S_k} \sum_{i \in P_S} \big| value(n,i) - value(n',i) \big| \quad - \\ & \sum_{k=1}^{K} \frac{1}{|S_k|} \left(1 - \frac{|P_D|}{|P|}\right) \\ & \sum_{n,n' \in S_k} \sum_{i \in P_D} \big( value(n,i) - value(n',i) \big)^2 \end{aligned} \tag{1}$$

**使得：**

$$|P_S| > 0 \tag{2}$$

$$|P_D| > 0 \tag{3}$$

$$P = P_S \cup P_D \cup P_U \tag{4}$$

$$P_S \cap P_D = \emptyset, P_U \cap P_D = \emptyset, P_S \cap P_U = \emptyset \tag{5}$$

$$\forall k \in [1,K], |S_k| > 1 \tag{6}$$

其中，目标函数的第一项计算 $P_S$ 中指标的显著性，第二项计算 $P_D$中指标的欧几里得距离。请注意，目标函数的第二项对应于 K-Means 方法所采用的经典聚类目标，$S = \{S_1, \ldots, S_K\}$ 是 N' 中顶点的簇，而 K 是一个常整数（即簇的数量）。

请注意，尽管公式（4）和（5）表示三划分的经典完整性约束，但公式（2）和（3）是必需的，以确保至少存在一个用于聚类（分别用于比较）的指标。

最后，在公式 (1) 中，使用术语 $\frac{|P_S|}{P}$ 和 $\frac{|P_D|}{P}$来平衡 $P_S$ 和 $P_D$之间的指标。

还可以添加更多约束来表示用户偏好，表明某个特定指标必须属于 $P_S$、$P_D$ 或 $P_U$。

由于该问题显然是非多项式的，我们将在下一节提出多种启发式方法来解决它。

## V. 算法解决方案

本节介绍我们提出的用于收集指标并提取比较洞见的算法。

### A. 收集指标

算法 1 详细说明了如何在给定节点类型的情况下，从属性图中获取一组有效指标。

**算法 1 带惰性验证的指标收集 要求：属性图 $G = (N, E, \rho, \lambda, \nu)$ 相对于图类型 $S = (N_S, E_S,$ 元素类型$_S, \rho_S)$ 有效，节点类型$n_S \in N_S$，阈值 $\alpha, \beta, \gamma$，要舍弃的属性名称集合 $D$ 确保：$n_S$ 的有效指标集 $I$ 1：$I = \emptyset$ 2：计算 $n_s$ 的上下文 $C_{n_S}$ 3：计算 $n_s$ 的指标集 $I_{n_S}$ 定义 9 4：对于 $i = \langle p, prop, type, agg \rangle \in I_{n_S}$，执行 5：如果 $|value(x_{prop}, prop)| \geq \gamma$，则 ▷ 可接受密度 6：如果 $i$与 $I$ 中的任何 $i'$ 都不相关，则 ▷ 非冗余 7：如果 $prop \notin D$ 且 $\alpha \leq |value(n_S, i)| \leq \beta$，则 ▷ 可接受方差 ▷ 和已舍弃属性 8：缩放$(i, value(n_S, i))$ ▷ 缩放 9：上下文化$(i, p)$ ▷ 上下文化 10：$I = I \cup i$ 11：结束 if 12：结束 if 13：结束 if 14：结束 for 15：返回 $I$**



算法 1 第 8 行和第 9 行使用的缩放与上下文化函数如下。

*a)* 缩放：我们注意到，缩放技术的选择对分类 [10] 和聚类 [33] 都很重要，其中百分位变换的表现最佳之一。下文中，我们使用百分位变换，其定义如下：对于$x$ 一个值和 $V$一个值集合，$percentileScaling(x, V) = \frac{|\{x' \in V | x' \leq x\}| \times 100}{|V|}$。

*b)* 上下文化：完成归一化后，我们根据路径的长度，为指标 $\langle path, prop, type, op \rangle$使用一个简单的衰减系数：$\frac{1}{1+k}$，其中 $p = (n, \ldots\ldots, e_k, n_k)$。

请注意，算法 1 会在获得候选指标集合后，才惰性地评估第 III-D 节中的验证属性。不过，有些验证属性可以尽早评估。例如，在根据属性设计指标时（算法 1 的第 2 行），即可检查可接受密度和已舍弃属性。因此，我们

实现了算法 1 的第二个版本，称为带急切验证的指标收集，该版本将可接受密度和已舍弃属性的检查提前到指标收集步骤。我们推测，这个较早舍弃属性的版本可能会加快指标收集和验证。

### *B.* 计算比较洞见

要解决第 IV 节中提出的问题，必须同时对指标和节点进行划分。可以通过二次整数规划或约束规划求得该问题的精确解，但对于包含数百个以上节点的数据集，这样做很可能难以处理 [2], [17]。本节将重点介绍实用启发式方法，即将问题拆分为两个优化阶段：首先划分指标，然后划分节点。这样做兼具两项优势：既能减少组合计算量，又能以较少的修改利用现有算法。我们首先介绍指标集的划分启发式方法。

*1*）指数启发式方法：第一种启发式方法会枚举已验证指标集的所有划分，然后针对每种划分求解节点划分问题。该启发式方法的复杂度为 $O(2^{|I_{ns}|})$，仅适用于包含少量指标的数据集，但预计能给出整体上最优的解。

*2*）基于拉普拉斯的启发式方法：算法 2 提出了一种简单的启发式方法，通过分析指标值，快速获得该问题的近似解。其思路是分别处理每个指标，并根据经典评分判断该指标适合归入哪个子集，从而决定将其用于分组（$P_D$）、比较（$P_S$）还是不使用（$P_U$）。

具体而言，我们使用拉普拉斯分数（LS） [18] 来评估该指标适合归入 $P_D$的程度，即评估其是否适用于聚类。在聚类中，LS 是一种无监督特征选择方法，它通过保持相似数据点之间的邻近关系，量化特征保留数据局部结构的能力。该方法利用特征值构建最近邻图，并计算图拉普拉斯矩阵 $L = D - A$，其中 $D$ 是图的对角度矩阵， $A$是用于表征图连通结构的相似度矩阵。

在我们的情形中，对于每个指标 $i$，给定节点类型 $n_S$，拉普拉斯分数的计算方式为： $LS(i) = \frac{I^T LI}{I^T DI}$其中$I = value(n_S, i)$ 是 $n_S$ 的指标值向量。

我们还使用变异系数 $CV$ 作为选择比较指标的多样性评分。给定节点类型 $n_S$ 的指标 $i$ ，其计算方式为 $CV(i) = \text{stdev}(value(n_S,i))\text{mean}(value(n_S,i))$。

**算法 2** 拉普拉斯启发式方法

**要求：** 节点类型 $n_S$ 的指标集合 $I$ with $values(n_S)$
**确保：** $P = \{P_S, P_D, P_U\}$ $\mathcal{I}$ 的一个三划分
1: $P_S = \emptyset, P_D = \emptyset, P_U = \emptyset$
2: **对** $i \in I$ **执行**
3: 计算 $LS(i)$
4: **</style}结束循环**
5: $Isort =$ 按 $LS(i)$ 对 $I$ 排序
6: $cut = elbow(Isort)$
7: **</style}对** $i \in I$ **执行**
8: **如果** $LS(i) < cut$ **则**
9: $P_D = P_D \cup \{i\}$ ▷用于分组
10: **结束如果**
11: **结束循环**
12: $I = I \setminus P_D$
13: **对** $i \in I$ **执行**
14: 计算 $CV(i)$
15: **结束循环**
16: $Isort =$ 按 $CV(i)$对 $I$ 排序
17: $cut = elbow(Isort)$
18: **对** $i \in I$ **执行**
19: **如果** $CV(i) < cut$ **则**
20: $P_U = P_U \cup \{i\}$ ▷ 未使用
21: **else**
22: $P_S = P_S \cup \{i\}$ ▷ 用于比较
23: **结束判断**
24: **结束循环**
25: **返回** $\{P_S, P_D, P_U\}$

算法 2 首先为每个指标计算局部搜索（第 3 行），按局部搜索值对指标排序（第 5 行），然后使用肘部检测对指标集进行划分（第 6 行），为 $P_D$ 保留得分最低的指标（第 7–11 行）。同样的原则也适用于其余指标（第 12 行），其中变异系数值（第 14 行）用于将它们划分（第 19 行）到 $P_U$（第 20 行）和 $P_S$ （第 22 行）中。

*3*）简单局部搜索：以上介绍的拉普拉斯启发式方法被用作局部搜索的起点，并以贪心方式使用 $P_S$、 $P_D$ 和 $P_U$ 之间的切换指标（见算法 3）。

更确切地说，令 $perm(P) = P'$ 表示一种置换操作，它将 $P = \{P_S, P_D, P_U\}$ 转换为新的 3-划分 $P' = \{P'_S, P'_D, P'_U\}$，使得 $|P_S \setminus P'_S| + |P_D \setminus P'_D| + |P_U \setminus P'_U| = 1$。算法 3 使用算法 2（第 1 行）计算问题的初始解 $P$。该 3-划分的质量通过公式 1 的目标函数得分来计算，即该函数给出的待最大化得分。随后，只要目标函数的增益高于用户指定阈值（第 4 行），算法就会继续迭代。每次迭代中，都会计算所有排列：从当前解的 $P$ 子集中交换两个指标（第 7 行），并选取增益最大的排列（第 9 至 13 行）用于下一次迭代。

*4*）节点划分的聚类启发式方法：在确定指标划分后，我们使用一种改进的聚类算法对节点集合进行划分。可以考虑两种方法。第一种方法是分别处理目标函数的第一项和第二项（见公式 1），从而得到两个节点划分，形成 $k$ 个簇。

**算法 3** 贪心局部搜索

**要求：** 一组 $I_{n_S}$ 指标，其中 $value(n_S) >$，一个目标函数上的阈值 old $\tau$ on the 目标函数

**确保：** $P = \{P_S, P_D, P_U\}$ $I$的三划分

```
P = 拉普拉斯启发式（I_nS， value(n_S)）
计算 obj 对于 P                ▷ 式 1 的目标函数
obj_last = 0
当 obj − obj_last ≥ τ 时执行          ▷当增益低于 τ 时停止
    obj_last = obj
    对所有排列 perm(P) = P′ 执行
        计算 obj_new 对于 P′      ▷目标函数公式 1
        如果 obj_new > obj_last 则
            obj = obj_new
            P_best = P′
        结束如果
    结束循环
    P = P_best                 ▷ new P 是最佳三划分
当.....时结束
返回 P
```

随后可以将这两个划分合并。这种方法称为共识聚类 [25] , 主要用于无法访问底层数据的情况。由于本例中的数据已知，我们选择计算单分区。我们采用改进版的模糊 C-中心点算法 [23]，在计算距离时同时考虑目标函数的两项（而不是像经典方法那样只考虑第二项）。与 K 均值等硬聚类算法相比，模糊聚类能够更好地处理离群点，并支持对簇中心解空间进行更渐进的探索。具体而言，模糊 C 均值会为每个实例分配其对各个簇的隶属度，并将其编码在隶属矩阵中。随后利用该矩阵计算簇中心的加权平均值，使优化过程能够更好地捕捉数据的底层结构。因此，与传统的硬聚类方法相比，损失函数中的聚类部分通常能够得到更精细的优化，但隶属矩阵会增加空间复杂度。

*5)* 随机 + 局部搜索启发式方法：最后，我们还修改了算法 3：从 $I$ 在 $P$中的指标随机分布开始，执行贪心局部搜索，对节点进行聚类，并重复此过程 5 次（即从 5 种不同的随机分布开始）。我们保留 5 个结果中依据公式 1 得分最高的解。

请注意，本节提出的算法解决方案会根据比较方案和分组方案计算节点簇，而不是直接计算比较洞见本身。基于得到的簇，可以按簇呈现比较洞见：给出一对节点，使比较方案中各指标的显著性达到最高。

| 数据库 | $\|N\|$ | $\|E\|$ | $\|N_S\|$ | $\|E_S\|$ | 平均属性数 N | 平均属性数 E |
|---|---|---|---|---|---|---|
| 机场 | 52,944 | 136,948 | 3 | 5 | 4.39 | 2.13 |
| ICIJ 泄密数据集 | 2,016,523 | 3,339,267 | 5 | 69 | 1.00 | 0.00 |
| 电影 | 28,863 | 166,261 | 6 | 4 | 1.60 | 1.20 |

表 IV：测试中使用的图数据集

## VI. 测试

### A. 实验设置

我们使用 Python 3.10 和 Neo4j 5.24，以及 APOC 5.24.2。在我们的实现中，收集候选指标的大部分计算工作都通过 Cypher 查询交由 Neo4j 完成。我们仅考虑数值属性来构建指标，并使用求和作为聚合函数。

*1)* 数据集：我们测试中使用的数据集汇总于表 IV。我们纳入了来自不同来源的真实图数据集。电影数据集是 Neo4j 示例仓库中的推荐图 [2]。ICIJ 泄密数据集来自国际调查记者同盟[3]。最后，为了测试节点具有许多数值属性的图，我们使用从 Kaggle.com 和 Eurostat[4], 收集的各种来源的数据，以及通过 fast-flights 库从 Google Flights 抓取的数据[5]，创建了一个 Neo4j 数据库（机场）。我们开发了一个 ETL 流程来统一和标准化数据，确保其一致性和完整性。对于这些数据库，节点或边中的所有数值属性均建立了范围索引。我们还纳入了一个经典的 UCI 数据集（Iris），因其规模非常小而选用（4 列，用于模拟指标；150 行，用于模拟待比较的节点）。

*2)* 实验方案：我们通过观察目标函数的得分（公式 1），评估算法在各数据集上的效率（运行时间，以秒为单位）以及不同启发式方法的质量。我们将启发式方法的效率和得分与随机 基线进行比较，其中指标在 $P_S, P_D$ 和 $P_U$ 之间随机划分。簇的数量在 $[2, 4]$ 中变化。

指标收集测试在一台 MacBook Pro 笔记本电脑上进行，该电脑配备 Apple Silicon M2 Pro 和 32GB 内存，运行 macOS 15.4。计算量更大的比较洞察提取实验则在一台 Fedora Linux 工作站上进行，该工作站配备两颗 2.3 GHz Intel Xeon 5118 12 核 CPU，以及 377GB 2666 MHz DDR4 主存。

### B. 收集指标的运行时间

我们首先分析算法 1 的两个版本——withlazy（pushdown=False）和 oreager（pushdown=True）验证——的效率。测试通过改变可接受密度阈值来收集指标，以此调整要丢弃的属性数量，并观察两种算法所耗费的时间

[2]https://github.com/neo4j-graph-examples [3] https://offshoreleaks.icij.org/ [4] https://ec.europa.eu/eurostat/web/main/home [5] https://aweirddev.github.io/flights/

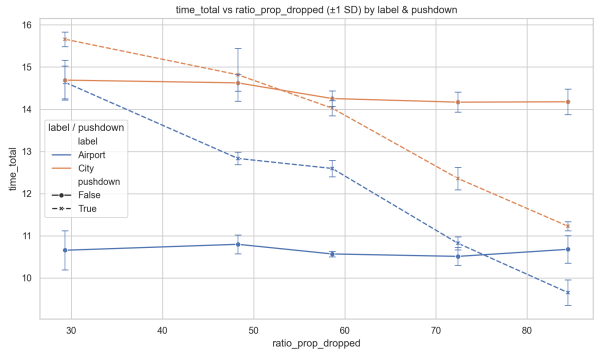



图 3：惰性验证与急切验证（总时间 (s)）

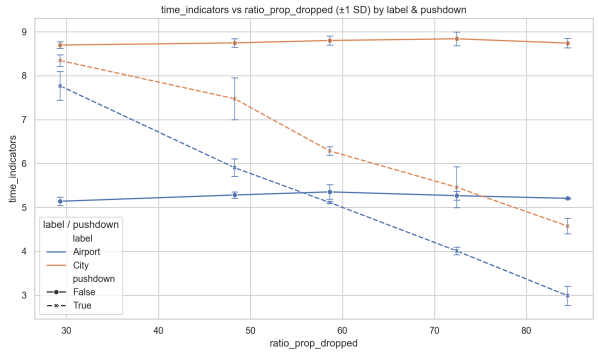



图 4：惰性验证与急切验证（候选指标收集时间（秒））

并对 3 次运行的结果取平均值。我们使用 Airports 数据库，并选择了两个标签（$n_S$ = Airport 或 City）来展示规模带来的影响（City 有 44,997 个节点，而 Airport 有 7,698 个节点）。从图 3 可以看出，丢弃的属性越多，急切验证的效率就越高。需要探索以收集指标的节点数量越多，这种影响就越显著。如图 4 所示，收集步骤的加速效果十分明显（算法 1 的第 2 行），因为该步骤依赖 Cypher 查询，下推会使查询更具选择性；而验证（第 4–9 行）则在主存中完成。

| | | | 时间（秒） | | | |
|---|---|---|---|---|---|---|
| 数据库 | $n_S$ | $\|IE(n_S)\|$ | 基数 | 候选项 | 验证 | 总计 |
| 机场 | 机场 | 7,698 | 0.8 | 5.1 | 2.3 | 10.4 |
| 机场 | City | 44,997 | 0.8 | 8.8 | 2.8 | 14.6 |
| 机场 | 国家 | 2,49 | 0.8 | 1.2 | 2.2 | 6.5 |
| ICIJ 泄密数据集 | 实体 | 814,344 | 94.7 | 210.4 | 58.4 | 364.3 |
| ICIJ 泄密数据集 | 中间实体 | 26,768 | 94.7 | 13.3 | 54.1 | 162.8 |
| ICIJ 泄密数据集 | 官员 | 771,315 | 94.7 | 236.9 | 56.9 | 389.2 |
| 电影 | 演员 | 15,443 | 0.8 | 5.4 | 3.6 | 10.5 |
| 电影 | 导演 | 4,091 | 0.8 | 1.6 | 3.4 | 6.6 |
| 电影 | 电影 | 9,125 | 0.8 | 2.7 | 3.5 | 7.8 |

表 V：收集指标的运行时间分解（惰性验证，可接受密度 = 80%）

表 V 详细列出了从 3 个图数据集的不同 $n_S$ 中获取 $values(n_S)$ 所需的时间（取 3 次运行的平均值），

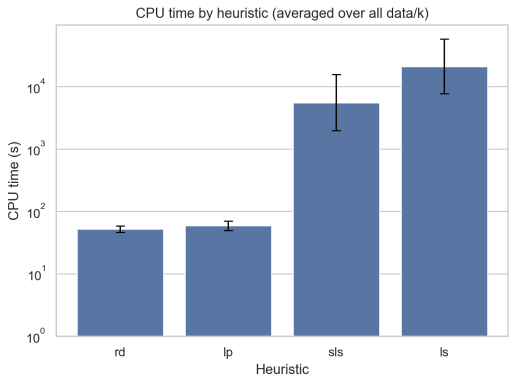



图 5：4 种启发式方法的 CPU 时间

也就是计算图中关系的基数，然后收集并验证指标（算法 1，采用惰性验证）。$IE(n_S)$ = Inst(ElemType($n_S$))。基数表示计算关系基数所需的时间，指标表示计算候选指标所需的时间，验证表示验证指标集所需的时间。

指标的验证采用以下标准：$\alpha$ 的可接受方差比为 $10^{-4}$ %，$\beta$ 为 100%，可接受密度为 80%，用于衡量非冗余性的皮尔逊相关系数为 $> 0.98$，并丢弃标识符。这些阈值是根据经验选取的，旨在去除无意义的指标。为确保能够计算显著性和距离，从 $values$（$n_S$）中删除了空值。

可以看出，正如预期，计算基数所需的时间取决于 $|E|$，即图中的边数（见表 IV）；而计算候选指标所需的时间主要受 $|IE$（$n_s$）$|$的影响。更令人意外的是，验证时间不仅取决于 $|IE$（$n_s$）$|$，还取决于 $|E|$（斯皮尔曼相关系数为 0.95）。这是因为 $|E|$越大，由于计算 $IE$（$n_s$）的新指标时会用到上下文，因此出现较大的 $|I_{n_S}|$的可能性也越高。

### C. 计算比较洞见的运行时间

| 数据集 | $n_S$ | 已验证数量 指标 | 已保留数量 节点 |
|---|---|---|---|
| Iris | - | 4 | 150 |
| 电影 | 导演 | 10 | 730 |
| 电影 | 电影 | 10 | 4143 |
| 机场 | 机场 | 28 | 3837 |

表 VI：用于洞察提取的指标集合

这组测试报告了在指标经过验证后提取比较洞察所需的时间。这些测试所使用的数据如表 VI 所述，比较的启发式方法包括：随机（*rd*）、拉普拉斯（*lp*）、简单局部搜索（*sls*）、随机 + 局部搜索（*ls*）。每项测试运行 10 次，并报告平均值和标准差。

图 5 报告了所有启发式方法的原始 CPU 时间，假设程序以单线程运行。不出所料，随机方法和

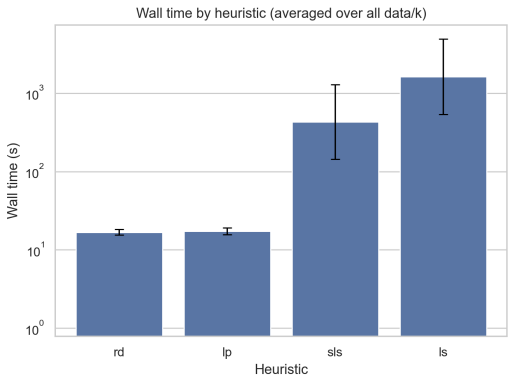



图 6：4 种启发式方法的墙钟时间

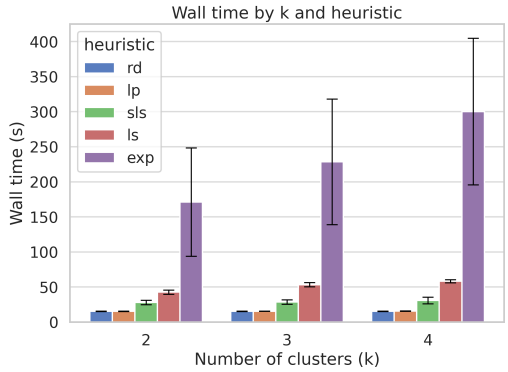



图 7：不同簇数量下的墙钟时间（Iris）

拉普拉斯启发式方法的速度最快，而两种局部搜索启发式算法都较慢，其中局部搜索 最慢，因为它需要反复搜索，以便更好地探索搜索空间。

我们还在图 6 中报告了使用多线程时所需的时间。可以看出，所有启发式方法都能利用多核处理器的优势，因为墙钟时间比 CPU 时间低了约一个数量级。

图 7 展示了所有启发式方法（包括指数算法 *exp*）在小型 Iris 数据集上的墙钟时间，并分析了期望簇数量 $k$ 的影响。可以看出，朴素启发式算法（随机、拉普拉斯）不受 $k$ 影响，而局部搜索、随机局部搜索 和 *exp* 会受到较大的 $k$ 影响。更重要的是，在这个小型数据集上，指数算法比局部搜索更快，这符合预期，因为搜索空间较小，而局部搜索需要进行固定次数的重复。作为比较，*exp*在 Movie 数据集上因 $k = 2$ 超时，耗时 10 小时。

### *D.* 找到的解的质量

在这最后一组测试中，我们分析所发现洞察的质量。这一质量通过目标函数（公式 1）所得的得分来表示。图 8 按数据集报告了归一化分数范围，并绘制了每种启发式方法获得的得分。归一化是通过汇总每种启发式方法在所有运行中针对某个数据集获得的全部得分，并使用最小-最大方法进行归一化来完成的。可以

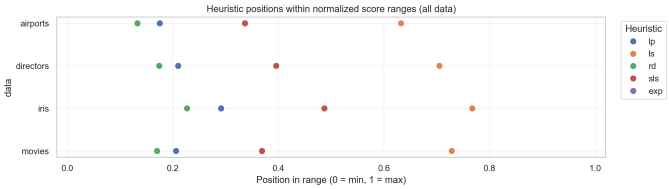



图 8：目标得分范围

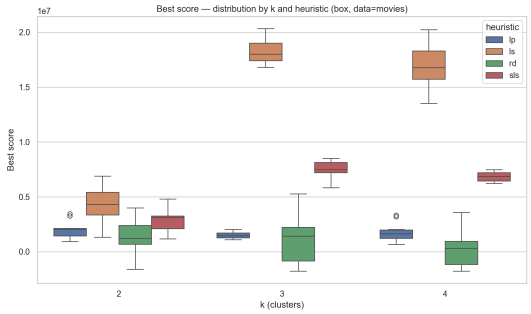



图 9：按启发式方法和簇数量划分的目标得分（电影）

看出，$ls$ 取得了最佳得分，这可以解释为它有 5 次机会通过局部搜索找到较优解。不出所料，随机方法和拉普拉斯启发式方法的得分最差，这也说明了使用局部搜索的必要性。

图 9 和图 10 展示了两种节点类型（Movies 数据库中的 Movie 和 Airports 数据库中的 Airport）按启发式方法和簇数量划分的得分分布。总体趋势是，较高的 $k$ 有利于获得更好的质量，但当 $k = 4$ 时似乎会达到平台期。图 11 展示了各启发式方法（包括指数方法 *exp*）在小型 Iris 数据集上获得的得分。可以看出，虽然 *exp* 在这个小型数据集上未能取得最佳得分（这是意料之中的），但它受这种平台期现象的影响较小，这表明随着 $k$ 增加，更充分地探索搜索空间具有重要意义。

## VII. 相关工作

本文将基于节点的上下文自动提取属性图中节点之间有意义的比较这一问题形式化

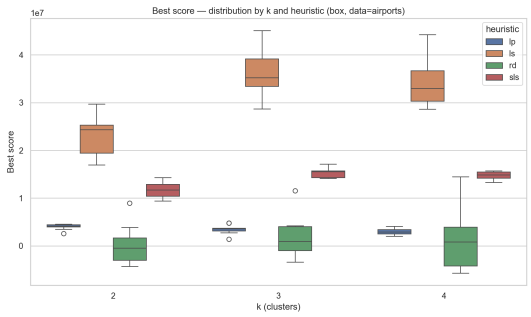



图 10：按启发式方法和簇数量划分的目标得分（Airport）

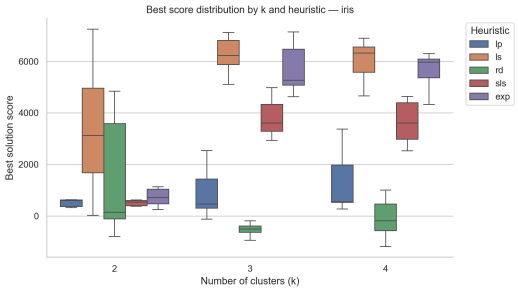



图11：不同启发式方法和簇数量下的目标得分（Iris）

，并提出了高效解决该问题的启发式方法。这使我们的工作归入探索性数据分析（EDA）领域。探索性数据分析通常是一项繁琐的任务，需要通过交互式分析数据集来获得洞见；这一领域近年来 [20], [24] 以及自早期发现驱动的多维数据探索以来 [30]，一直备受关注。自动生成洞见以支持探索性数据分析的方法通常分为两类：生成并筛选方法 [12], [32],会枚举潜在洞见并为其排名，而引导式探索性数据分析 方法 [14], [27], 则以类似人类的方式迭代探索搜索空间。我们的方法属于前一类，侧重于根据上下文相关性生成有意义的节点间比较并确定其优先级。

多项研究强调了比较在数据分析中的重要性 [6], [34]。已有方法可用于提取比较洞见 [8] 或表达比较查询 [31],，但据我们所知，这些方法仅适用于关系型数据库。与之不同，我们的工作面向属性图上的分析任务。

图分析通常指超越基础联机事务处理查询，对图数据进行分析和丰富，并利用结构模式开展高级分析 [7], [9]。虽然 PageRank 和 ShortestPaths 等经典算法仍是 LDBC Graphalytics 等基准测试的核心 [21],，近期研究则侧重于扩展由应用驱动的功能，使用户能够执行复杂分析。例如，近期研究在 Cypher 中引入了流式运算符 [29], 并在 GQL 中引入了因果推断构造 [26]。然而，这些研究的目的与我们的工作不同。

其他研究，例如 [13], ，探索属性图分析，以估算正则路径查询的计数。其目标是将节点聚合为超节点，从而生成诸如内部顶点数量等统计属性，而非考察各个节点的属性来生成指标。 [28] 研究了一种领域无关的 RDF 图实体比较方法，通过查询计算相似之处和差异，而不依赖显式路径。虽然该研究生成单个 SPARQL查询来解释实体相似性，但我们的工作会推导出一组指标，依据属性和拓扑对属性图进行比较。

我们的方法采用多阶段、基于过滤的流程，并通过启发式搜索加以优化，根据特征的内在属性对其进行评估。该流程包含两个阶段：第一阶段专门用于设计指标，第二阶段用于计算洞见。第一阶段遵循特征工程原则，因为比较指标充当特征，用于捕捉图的结构和语义特征。特征工程技术旨在缩小搜索空间并提升模型性能，从而缓解在高维图中生成比较洞见时遭遇的维度灾难 [5], [11], [22]。尽管我们工作的重点——节点比较洞见——是一项创新贡献，但我们的方法建立在成熟实践的基础上，采用基于过滤的选择来设计指标，并结合缩放和衰减技术。

第二阶段采用成熟的启发式优化技术，将拉普拉斯启发式方法与随机和局部搜索策略相结合 [11]，[18]，[22]。这种方法利用拉普拉斯的结构属性来引导搜索过程，同时结合随机搜索和局部搜索技术的探索与微调能力 [19]。由此得到的解决方案无需穷举搜索空间，即可计算洞见，即使面对大型图也是如此。其他同样采用基于启发式方法来优化搜索空间、但目标函数与我们不同的相关研究包括：从知识库自动生成比较表 [16], 、自动进行分面生成 [15],, 以及自动推导知识图谱的排名指标 [1]。

我们工作的一个关键创新之处在于，基于节点上下文为属性图定义指标，同时对问题进行形式化，并提出若干解决该问题的方法。

## VIII. 结论

本文提出了一种从属性图中提取节点比较洞见的方法。比较洞见依赖于通过识别待比较节点的上下文而构建的指标。我们对如何选择用于对节点分组的指标，以及如何在组内比较节点的问题进行了形式化，并提出了多种启发式方法，以获得该问题的近似解。

我们的短期计划包括调整当前方法，例如引入用户偏好来设计指标、以更细粒度的层级处理验证属性，或利用全局属性（如节点中心性度量）。从长远来看，我们计划研究属性图特有的其他类型的比较洞见（例如路径比较），并寻找自动为属性图补充额外数值属性的方法，使其更适合提取比较结果。

## IX. AI 生成内容声明

GPT5 已用于实现实验中使用的部分算法和 Cypher 查询（参见第五节和第六节）。我们的代码可在 Git 仓库中获取：https: //github.com/AlexChanson/Comparing-Nodes。

# 参考文献


[1] H. Abdallah、B. Markhoff 和 A. Soulet。从超大型知识图谱中发现排名指标。《VLDB Endowment 会议录》，18(4):1183–1195，2024。

[2] K. C. ´Agoston 和 M. E.-Nagy。k 均值聚类问题的混合整数线性规划建模。《中欧运筹学杂志》，32(1):11–27，2024 年 3 月。

[3] R. Angles、A. Bonifati、S. Dumbrava、G. Fletcher、A. Green、J. Hidders、B. Li、L. Libkin、V. Marsault、W. Martens、F. Murlak、S. Plantikow、O. Savkovic、M. Schmidt、J. Sequeda、S. Staworko、D. Tomaszuk、H. Voigt、D. Vrgoc、M. Wu 和 D. Zivkovic。Pg-schema：属性图的模式。《ACM 管理数据会议录》，1(2):198:1–198:25，2023。

[4] R. Angles、A. Bonifati、R. Garc´ıa 和 D. Vrgoc。图查询语言的基于路径的代数基础。载于 A. Simitsis、B. Kemme、A. Queralt、O. Romero 和 P. Jovanovic（编），第 28 届数据库技术扩展国际会议论文集，EDBT 2025，西班牙巴塞罗那，2025 年 3 月 25—28 日，第 783–795 页。OpenProceedings.org，2025。

[5] M. C. Barbieri、B. I. Grisci 和 M. Dorn。面向性能和稳定性的特征选择方法分析与比较。《专家系统与应用》，249:123667，2024。

[6] T. Blount、L. Koesten、Y. Zhao 和 E. Simperl。了解新手数据讲述者对叙事模式的运用。载于 CHIRA，第 128–138 页，2020。

[7] A. Bonifati、M. T. ¨Ozsu、Y. Tian、H. Voigt、W. Yu 和 W. Zhang。图分析路线图。SIGMOD Rec.，53(4):43–51，2024。

[8] A. Chanson、N. Labroche、P. Marcel、S. Rizzi 和 V. T'kindt。用于交互式数据探索的比较笔记本自动生成。载于 J. Stoyanovich、J. Teubner、P. Guagliardo、M. Nikolic、A. Pieris、J. M¨uhlig、F. ¨Ozcan、S. Schelter、H. V. Jagadish 和 M. Zhang（编），第 25 届数据库技术扩展国际会议论文集，EDBT 2022，第 1 卷，2022 年，29 2 274 2 284 2022 年 3 月至 4 月，英国爱丁堡，页：–：。OpenProceedings.org，。

[9] U. Cheramangalath、R. Nasre 和 Y. Srikant。图分析导论。载于《分布式图分析：编程、语言及其编译》，第 1–29 页。Springer，2020。

[10] L. B. V. de Amorim、G. D. C. Cavalcanti 和 R. M. O. Cruz。缩放技术的选择对分类性能至关重要。《应用软计算》，133:109924，2023。

[11] P. Dhal 和 C. Azad。机器学习各领域中特征选择的综合综述。《应用智能》，52 4581，2022 4 4543( ): – 。

[12] R. Ding、S. Han、Y. Xu、H. Zhang 和 D. Zhang。QuickInsights：从多维数据中快速、自动地发现洞察。载于 SIGMOD，第 317–332 页，2019。

[13] S. Dumbrava、A. Bonifati、A. N. R. Diaz 和 R. Vuillemot。属性图上的近似查询。载于可扩展不确定性管理国际会议，第 250–265 页。Springer，2019。

[14] O. B. El、T. Milo 和 A. Somech。使用深度强化学习自动生成数据探索会话。载于 SIGMOD，第 1527–1537 页，美国俄勒冈州波特兰，2020。

[15] L. Feddoul 、 S. Schindler 和 F. L¨offler。知识图谱上的自动分面生成与选择。载于语义系统国际会议，第 310–325 页。Springer 国际出版公司，瑞士沙姆，2019。

[16] A. Giacometti、B. Markhoff 和 A. Soulet。基于知识库的比较表生成。载于 R. Verborgh、K. Hose、H. Paulheim、P. Champin、M. Maleshkova、 ´O. Corcho、P. Ristoski 和 M. Alam（编），《语义网——第 18 届国际会议》，ESWC 2021，虚拟活动，2021 年 6 月 6—10 日，论文集，《计算机科学讲义》第 12731 卷，第 179–194 页。Springer，2021。

[17] S. Gilpin、S. Nijssen 和 I. Davidson。将层次聚类形式化为整数线性规划。《AAAI人工智能会议论文集》，27(1):372–378，2013 年 6 月。

[18] X. He、D. Cai 和 P. Niyogi。用于特征选择的拉普拉斯分数。载于《神经信息处理系统进展》 *18 [*神经信息处理系统，NIPS 2005，2005 年 12 月 5—8 日，加拿大不列颠哥伦比亚省温哥华*]*.第 507–514 页，2005。

[19] J. Hromkovic。《难题算法学——组合优化、随机化、近似与启发式方法导论》。Springer，2001。

[20] S. Idreos、O. Papaemmanouil 和 S. Chaudhuri。数据探索技术概述。载于 SIGMOD，第 277–281 页，2015。

[21] A. Iosup, T. Hegeman, W. L. Ngai, S. Heldens, A. Prat-P´erez, T. Man- hardto, H. Chafio, M. Capot˘a, N. Sundaram, M. Anderson 等。LDBC Graphalytics：面向并行和分布式平台的大规模图分析基准。*VLDB* 会刊，9(13):1317–1328，2016。

[22] M. R. Islam, A. A. Lima, S. C. Das, M. F. Mridha, A. R. Prodeep 和 Y. Watanobe。关于特征选择的流程、方法、评估及挑战的全面综述。*IEEE Access*，10:99595–99632，2022。

[23] R. Krishnapuram, A. Joshi, O. Nasraoui 和 L. Yi。用于网络挖掘的低复杂度模糊关系聚类算法。*IEEE* 模糊系统汇刊，9(4):595–607，2001。

[24] T. Milo 和 A. Somech。通过机器学习自动执行探索性数据分析：概述。载于*SIGMOD*，第 2617–2622 页，2020。

[25] N. Nguyen 和 R. Caruana。共识聚类。载于*2007* 年第七届 *IEEE* 国际数据挖掘会议论文集，ICDM '07，第 607–612 页，美国，2007。IEEE 计算机学会。

[26] A. Pachera, M. Palmiotto, A. Bonifati 和 A. Mauri。假如：使用图数据库进行因果分析。*VLDB* 会刊，18(11):4009–4016，2025 年 9 月。

[27] A. Personnaz, B. Youngmann 和 S. Amer-Yahia。Eda4sum：数据摘要的引导式探索。*VLDB* 会刊，15 3593，2022 12 3590( ): – 。

[28] A. Petrova, E. Sherkhonov, B. Cuenca Grau 和 I. Horrocks。RDF 图中的实体比较。载于 C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudr´ e-Mauroux, J. Sequeda, C. Lange 和 J. Heflin 编，语义网——*ISWC 2017*，第 526–541 页，Cham，2017。Springer 国际出版公司。

[29] C. Rost, R. Tommasini, A. Bonifati, E. D. Valle, E. Rahm, K. W. Hare, S. Plantikow, H. Voigt 和 P. Selmer。Seraph：属性图流上的连续查询。载于*EDBT/ICDT 2025* 联合会议，法国帕埃斯图姆，2024 年 3 月。

[30] S. Sarawagi, R. Agrawal 和 N. Megiddo。OLAP 数据立方体的发现驱动探索。载于*EDBT*，第 168–182 页，1998。

[31] T. Siddiqui, S. Chaudhuri 和 V. R. Narasayya。COMPARE：加速关系型数据库中的分组比较，以实现数据分析。*VLDB*基金会，14(11):2419–2431，2021。

[32] B. Tang, S. Han, M. L. Yiu, R. Ding 和 D. Zhang。从多维数据中提取排名前 k 的洞见。载于 S. Salihoglu, W. Zhou, R. Chirkova, J. Yang 和 D. Suciu 编，*2017* 年 *ACM* 国际数据管理会议论文集，*SIGMOD 2017* 会议，*2017* 年 *5* 月 *14–19* 日，美国伊利诺伊州芝加哥，第 1509–1524 页。ACM，2017。

[33] C. Wongoutong。忽略 k 均值聚类中的特征缩放所造成的影响。*PLOS ONE*，19(12):e0310839，2024。

[34] E. Zgraggen, Z. Zhao, R. C. Zeleznik 和 T. Kraska。研究多重比较问题对可视分析的影响。载于*CHI*，第 479 页，2018。