动态完全符号图的相关聚类算法：基于索引的方法

Ali Shakiba

拉夫桑贾恩瓦利阿斯尔大学计算机科学系，伊朗拉夫桑贾恩。
ali.shakiba@vru.ac.ir;a.shakiba.iran@gmail.com

arXiv:2301.00384v1 [cs.DS] 1 Jan 2023

摘要

本文将相关聚类问题的近似复杂度从$\mathcal{O}(m \times (2 + \alpha(G)) + n)$降至 $\mathcal{O}(m + n)$，适用于任意给定的 ε 值；对于一个含有 n 个顶点和 m 条正边的完全符号图，其中 $\alpha(G)$为图的森林度。我们的方法能够给出与原始算法相同的输出，并使该算法能够在允许边符号翻转和顶点添加/删除的完全动态设置下实现。构建此索引需要 $\mathcal{O}(m)$内存和 $\mathcal{O}(m \times \alpha(G))$时间。我们还研究了近似算法中所用非一致性度量的结构性质。理论结果辅以针对七个真实世界图的完整实验。结果表明，与非索引算法相比，我们的基于索引的算法平均可将运行时间缩短34%。

关键词： 相关聚类 · 动态图 · 在线算法

1 引言

聚类是机器学习领域中研究最广泛的问题之一，在大型数据集的分析与可视化等方面有着各种应用。人们提出了多种模型和技术来对元素进行划分，使得不同划分中的元素彼此不相似，而同一划分中的元素彼此非常相似。

文献 [1]，中提出的相关聚类问题，在不一致最小化方面已知是一个 **NP**-难问题。因此，文献中存在多种基于其整数规划表述的不同近似解。最近，文献 [1] 中的 2-近似算法思路在文献 [4] 中得到了扩展，用于构造一个 $\mathcal{O}(1)$-近似算法。文献 [4] 中的实验表明，该算法在实际应用中性能尚可，尽管其理论保证可能很高，例如文献 $\beta = \lambda = \frac{1}{36}$ 中的 1 442。在文献 [3], 中，该算法被扩展到在线设置，其中只允许添加顶点；每当添加一个新顶点时，它会揭示其所有正符号边。Shakiba 在文献 [12] 中研究了底层图中的顶点添加/删除和边符号翻转对最终聚类结果的影响，以使该算法适用于动态图。然而，这种方法的一个瓶颈在于计算边之间的 NonAgreement 值，并确定顶点的 ε-轻度。本文提出了一种新颖的索引方案来解决这一问题，使算法高效运行，不仅适用于动态图，也适用于动态超参数 ε。与文献 [3] 中的在线方法相比，我们所提方法允许完全动态设置，即顶点添加/删除以及边符号翻转。已知任何用于相关聚类问题的在线算法，其近似比至少为 $\Omega(n)$ [10]。请注意，本文使用的底层算法是一致的，这一点已通过实验结果 [3] 得到证实。

本文其余部分安排如下：第 1.1 节介绍我们的贡献。随后，第 2 节回顾一些基本算法和结果。接着，我们在第 3.1 节介绍新颖的索引结构，并展示如何利用该结构提升近似相关聚类算法的运行时间。然后，我们在第 3.2 节展示如何在完全动态设置下维护所提出的索引。第 4 节中，我们进行了大量实验，以验证这些

理论结果，并展示所提出的索引结构的有效性。最后，本文给出结论。

1.1 我们的贡献

本文探讨的问题很简单：

“如何缩短近似求解输入图 [4]在不同 ε取值下的相关聚类所需的时间？”

我们还想知道：

“How 能否将第一个问题的解转化为适用于动态图的在线解？ ”

对于第一个问题，我们的答案是设计一种新颖的索引结构，该结构基于近似算法的结构性质及其 NonAgreement 度量构建。正如第 4 节的实验所示，对于七个真实世界数据集，所提方法平均将查询聚类的总运行时间缩短了约 %34。随后，我们依据 [12]中的理论结果，使该结构能够在线处理动态图。索引本身的构建具有很高的并行化潜力，最多可并行到输入图中的顶点数。并行化的思路很简单：使用独立的并行线程，在 $\mathbb{NAO}$（G）中分别构建每个$\mathbb{NAO}$（v）。我们还研究了NonAgreement度量中的内在结构，以便设计更高效的算法，在底层图发生更新时维护索引。更具体地说，我们证明，使用所提索引结构，对于任意给定的 ε 值，都可以在 $\mathcal{O}$（$m+n$）时间内求得图的一个相关聚类；相比之下，CC 需要 $\mathcal{O}$（$m\times$（$2+\alpha$（G））$+n$）时间。ICC 的预处理时间为 $\mathcal{O}$（$m\times\alpha$（G））， 空间复杂度为 $\mathcal{O}$（m）。

2 预备知识

令 $G=(V,\ E)$ 为一个具有 $|V|=n$ 个顶点的完全无向符号图。边的集合 E 自然划分为正符号边和负符号边，分别记为 E^+ 和 E^-。随后，我们使用 m 表示$|E^+|$。相关聚类问题定义为

$$\mathrm{cost}(\mathcal{C})=\sum_{\substack{\{u,v\}\in E^+\\ u\in C_i,v\in C_j,i\neq j}}1+\sum_{\substack{\{u,v\}\in E^-\\ u,v\in C_i}}1, \tag{1}$$

其中，$\mathcal{C}=\{C_1,\ldots,C_\ell\}$ 是一个聚类。注意，这是该问题的最小分歧变体。

相关聚类问题的常数因子近似算法 [4]基于两个主要量：（1）正符号边 $\{u,v\}$ 的 ε-一致性，即当且仅当 u 和 v 处于 ε-一致状态时，$\textsc{NonAgreement}_G(u,v)=|N_G(u)\Delta N_G(v)|\ \mathsf{max}\{|N_G(u)|,\ |N_G(v)|\}<\varepsilon$；以及（2）$\varepsilon$-轻量性，其中，如果 $\frac{\textsc{AgreeCnt}_{G^+}(u)}{|N_{G^+}u|}<\varepsilon$，则称顶点 u 为 ε-轻量，其中$\textsc{AgreeCnt}_{G^+}(u)=|\{w\in V|u$ 且 v 与 ε 处于一致状态$\}|$。注意，不是 ε-轻量的顶点称为ε-重顶点。这是一个 $2+\frac{4}{\varepsilon}+\frac{1}{\varepsilon^2}$-近似算法，相关证明见 [3]。该算法见算法 1，简而言之，我们将其称为 CC 算法。

[12] 中的 Shakiba 研究了完全动态设置下 CC 算法的理论基础。以下结果总结了 [12] 中的表 1、推论 1 和定理 4。

定理 1。 假设边 $u=\{u,v\}$ 的符号发生翻转。那么，除与 u 和 v 中任一顶点的距离大于二的顶点以外，其他顶点的非一致性和 ε-轻度均不会发生变化。

图 G 的森林度是将其分解为边不相交生成森林所需的最少数量。以下关于森林度的引理有助于界定操作次数。

引理 1。 *[2]* 中的引理 *2*：设图 $G=(V,E)$ 有 n 个顶点和 m 条边。那么，

$$\sum_{\{u,v\}\in E}\min\{\deg_G(u),\deg_G(v)\}\leq 2a(G)\times m. \tag{2}$$

算法 1 CorrelationClustering(G) [4]

1：过程 CorrelationClustering(G, ε) **2：令** $G^+ = G[E^+]$ **，其中** E^+ **是符号为** $+$ **的边的集合 3：丢弃所有端点不在** ε**-一致性范围内的边 4：丢弃两个** ε**-轻顶点之间的所有边 5：令** G^+ **为执行前述两项操作后的稀疏图** G^+ **6：令** $\mathcal{C}$ **为** G^+ **中连通分量的集合 7：返回** $\mathcal{C}$ **作为输出聚类 8：结束过程**

3 所提方法

本节将介绍我们新颖的索引结构。该结构支持针对动态图中不同 ε 值的相关聚类进行动态查询。采用该索引结构的算法称为 ICC，即基于索引的相关聚类。

3.1 索引结构

对于符号为正号的边 $e = \{u, v\}$，我们将其 ε-一致性距离 定义为 $\text{NonAgreement}_{G^+}(u, v)$。直观地说，这是使节点 u 和 v 不再处于 ε-一致状态的值 ε 的下确界。令集合 $\mathcal{E} = \{\text{NonAgreement}_{G^+}(u, v)$ 为 $|e = \{u, v\} \in E^+\}$。不失一般性，设 $\mathcal{E} = \{\varepsilon_0, \ldots, \varepsilon_{\ell-1}\}$，且排序满足 $\min \mathcal{E} = \varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_{\ell-1} = \max \mathcal{E}$。对于固定的 ε 值，令 $G^+_\varepsilon = (V, E^+_\varepsilon)$，其中 $E^+_\varepsilon = \{e = \{u, v\} \in E^+ | \text{NonAgreement}_{G^+}(u, v)$ 为 $< \varepsilon\}$。

观察 1. 对于所有 $\varepsilon \le \varepsilon_0$，$G^+_\varepsilon$ 是空图，即包含所有节点但不含任何边的图。此外，对于所有 $\varepsilon > \varepsilon_\ell$，$G^+_\varepsilon = G^+$。

接下来，我们介绍索引结构的关键要素，称为 $\mathbb{NAO}$。

定义 1（NonAgreement 节点排序）**.** 每个节点 $v \in V$ 的 ε-一致排序，记为 $\mathbb{NAO}(v)$，定义为 G 中的一个有序顶点子集，其中：

1. 当且仅当 $e = \{u, v\}$ 是 G 中的一条正边时，节点 $u \in V$ 才会出现在排序 $\mathbb{NAO}(v)$ 中。
2. 对于出现在 $\mathbb{NAO}(v)$ 中的任意两个不同顶点 $u, w \in V$，

$$\text{NonAgreement}_{G^+}(v, u) < \text{NonAgreement}_{G^+}(v, w), \tag{3}$$

意味着 u 出现在 w 之前。

3. 对于每个节点 $u \in \mathbb{NAO}(v)$，该节点还存储其 ε-一致距离 .

图 G 的 NonAgreement 节点排序定义为 $\mathbb{NAO}(G) = \{(v, \mathbb{NAO}(v)) | v \in V\}$。

换言之，$\mathbb{NAO}(v)$ 是 v 在 G^+ 中按其 ε-一致距离值排序后的邻居节点数组。示例图中所有顶点的一个示例 NonAgreement 节点排序如图1所示。接下来的两个引理将分析 $\mathbb{NAO}(G)$ 的空间和构建时间复杂度。

引理 2. 图 G, $\mathbb{NAO}(G)$ 的 NonAgreement 节点排序可用 $\mathcal{O}(m)$ 内存表示。

证明。 $\mathbb{NAO}(v)$ 内节点的数量等于所有顶点 $N_{G^+}(v)$ 以及顶点 v 本身的度数 $_{G^+}(v) + 1$。请注意，无需在排序中显式存储顶点 v 本身。总而言之，表示 $\mathbb{NAO}(v)$ 所需的总空间为 $2 \times m$ 个条目。

□

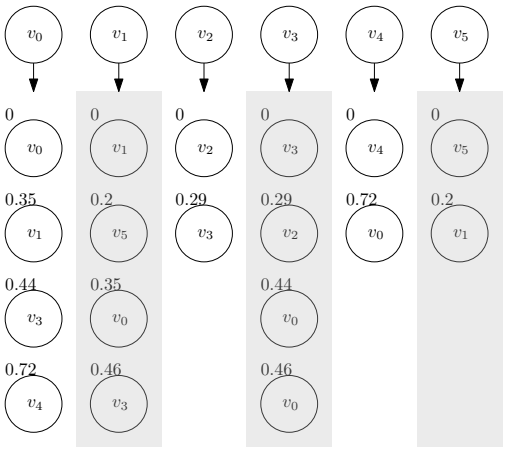


图1：一个示例图中 $\mathbb{NAO}(v)$ 的示意图。

引理 3。为所有顶点 $v \in V$ 构建 $\mathbb{NAO}(v)$ 的时间复杂度为 $\mathcal{O}(m \times (\alpha(G) + \lg m))$，其中 $\alpha(G)$ 是图 G的森林度。

证明。要计算所有边 $e = \{u, v\} \in E^+$ 的 $\textsc{NonAgreement}_{G^+}(u, v)$ 的值，需要计算 $|N_{G^+}(u) \Delta N_{G^+}(v)|$。假设 u 和 v 的邻接表均已排序，则计算对称差需要进行 $\mathcal{O}(\deg_{G^+}(u) + \deg_{G^+}(v))$ 次操作。不过，我们可以先计算它们的交集，并据此计算 $\textsc{NonAgreement}$，如下所示

$$\textsc{NonAgreement}_{G^+}(u, v) = \frac{\deg_{G^+}(u) + \deg_{G^+}(v) - 2 \times |N_{G^+}(u) \cap N_{G^+}(v)|}{\max\{N_{G^+}(u), N_{G^+}(v)\}}. \tag{4}$$

因此，计算 $\textsc{NonAgreement}$ 的值所需的总时间等于 t o

$$\sum_{\{u,v\} \in E^+} \min\{\deg_{G^+}(u), \deg_{G^+}(v)\},$$

已知其上界为 $2\alpha(G) \times m$（引理 1）。此外，每个 $\mathbb{NAO}(v)$ 的长度为 $1 + \deg_{G^+}(v)$，因此需要按其 $\textsc{NonAgreement}$ 的值进行排序，耗时为 $\mathcal{O}(\deg_{G^+}(v) \lg(\deg_{G^+}(v)))$；总计耗时为 $\mathcal{O}(m \lg m)$。

□

给定 ε 和图 G 所对应的相关聚类，是图 G^+ 的连通 ⁀ 分量集合。给定对 $\mathbb{NAO}(G) = \{\mathbb{NAO}(v) | v \in V(G)\}$ 的访问权限，可以回答以下查询：(1) 顶点 v 是 ε-重顶点还是 ε-轻顶点？(2) 边 $\{u, v\}$ 的两个端点是否处于 ε-一致状态？正如本节将要展示的，这两个问题都可以利用 $\mathbb{NAO}$ 结构在 $\mathcal{O}(1)$ 时间内得到解答。对于顶点 $v \in V$，其 ε-轻阈值定义为 $\mathbb{LTH}_\varepsilon(v) = \lceil \varepsilon \deg_{G^+}(v) \rceil + 1$。

引理 4。对于一个 ε 和一个顶点 $v \in V$，当且仅当 $\mathbb{LTH}_\varepsilon(v)$ 中第 ε 小的顶点在 $\mathbb{NAO}(v)$ 中的 ε-一致距离小于 ε 时，v 才是 ε-重顶点。否则，它是 ε-轻顶点。

证明。</*style*当且仅当 (，v) $< \varepsilon$ 时，顶点 u 和 v 才会处于 ε-一致状态。每当 $\mathbb{LTH}_\varepsilon(v)$ 中第 v 个最小顶点的 ε-一致距离小于 ε 时，这意味着至少有 $\lceil \varepsilon \deg_{G^+}(v) \rceil + 1$ 个顶点（包括 v 本身）与 v 处于 ε-一致状态。这等价于顶点 v 的 ε-重性。另一方面，如果 $\mathbb{NAO}(v)$ 中第 v 个最小顶点的 ε-一致距离大于或等于 ε，则与 v 处于 ε-一致状态的顶点数量小于 $\varepsilon \times \deg_{G^+}(v)$，也就是说，$v$ 是 ε-轻的。

□

引理 5. 在已知 $\mathbb{NAO}(v)$ 的情况下，对于所有 ε 的取值以及任意顶点 $v \in V$，可以在 $\mathcal{O}(1)$ 内确定 v 的 ε-轻度。

证明。这可由引理 4 推得，前提是将 $\mathbb{NAO}(v)$ 实现为数组。 □

给定对图 G 的索引 $\mathbb{NAO}(G) = \{(v, \mathbb{NAO}(v))|v \in V\}$ 的访问权限，查询只需针对给定的 ε 值，求出该图的一个聚类。

定理 2. 给定对索引 $\mathbb{NAO}(G)$ 的访问权限，可在 $\mathcal{O}(m(a(n)+1))$ 摊销时间内，针对给定参数 ε 计算一个聚类，其中 $a(n)$ 是单值阿克曼函数的缓慢增长的反函数。

~~证明。~~ 直观地说，我们将使用 $\mathbb{NAO}(G)$ 构造图 G^+，并逐步计算其连通分量。更正式地说，我们首先将每个顶点放入并查集结构中各自独立的集合。然后，我们找出 ε-重顶点 $\mathcal{H} \subseteq V$，这需要 $\mathcal{O}(m)$。接下来，考虑 $\mathbb{NAO}(v)$，其中$v \in \mathcal{H}$。对于每个键小于 ε 且满足 $u \in \mathcal{H}$ 的顶点 $u \in \mathbb{NAO}(v)$，我们调用 $\textsc{Union}(u, v)$，合并包含 u 和 v 的簇。这些操作使用并查集数据结构需要 $\mathcal{O}(m \times a(n))$ 摊销时间 [5]。查询算法的正确性由定义 1 以及引理 4 和 5 推出。

□

在进一步讨论之前，我们需要陈述以下结果，它们直观地考察了通过 $\mathcal{E}$，边和顶点的 ε-一致性及 ε-轻量性表现出的行为。令

$$\mathcal{E} = \left\{\textsc{NonAgreement}_{G^+}(u,v) \,|\, \{u,v\} \in E^+\right\} = \{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{\ell-1}\}, \tag{5}$$

其中 $\varepsilon_{i-1} < \varepsilon_i$对于 $i = 1$、.....、$\ell - 1$。此外，假设 $\varepsilon_\ell > \max \mathcal{E}$。

观察 2。 对于任意 $\varepsilon \leq \varepsilon_0$，所有正号边 $\{u, v\}$ 的端点都不处于 ε-一致状态。

观察 3。 对于任意 $\varepsilon > \varepsilon_{\ell-1}$，所有正号边 $\{u, v\}$ 的端点都处于 ε-一致状态。

根据观察 2，对于所有 $\varepsilon \leq \varepsilon_0$ 的取值，算法 1 得到的相关聚类结果都是由单点顶点组成的集合，即 $\{\{v\}\,|v \in V\}$。然而，对于 $\varepsilon > \varepsilon_{ell-1}$，我们不能断言输出是单个簇，即集合 V。为什么呢？随着 ε 增大，处于 ε-一致状态的边的数量会增加，但 ε-轻量顶点的数量也会增加（依据即将给出的推论 1）。因此，要选择合适的 ε 以获得最少的簇数量，就需要权衡。我们将在第 4 节（实验）中进一步讨论这一问题。

定理 3. 设 $\varepsilon < \varepsilon'$ 和 $u, v \in V$ 是两个不同的顶点。如果 u 和 v 处于 ε-一致状态，那么它们也会处于 ε'-一致状态。此外，如果 u 和 v 不处于 ε'-一致状态，那么它们也不会处于 ε-一致状态。

证明. 该证明可由 $\textsc{NonAgreement}$ 定义直接推出。由于 u 和 v 处于ε-一致状态，因此有 $\textsc{NonAgreement}_{G^+}(u,v) < \varepsilon < \varepsilon'$，即 u和 v 也处于 ε'-一致状态。同理，由于 u和 v 不处于 ε'-一致状态，因此有 $\textsc{NonAgreement}_{G^+}(u,v) \geq \varepsilon' > \varepsilon$，即 u 和 v 也不处于 ε-一致状态。 □

定理 4. 设 $\varepsilon < \varepsilon'$ 和 $u \in V$。如果 u 是 ε-轻的，那么 u 也会是 ε'-轻的。此外，如果 u 是 ε'-重的，那么它会是 ε-重的。

证明。 根据 ε-轻量性的定义和定理 3，可得此结论。 □

定理 4 未考虑另外两个重要情形，即：(1) 顶点 u 是否可能是 ε-重型的，但在某些 $\varepsilon < \varepsilon'$ 值下变为 ε'-轻型？(2) ε-轻型顶点是否可能在某些 $\varepsilon < \varepsilon'$ 值下变为ε'-重型？这两个问题的答案都是肯定的。根据定理 3 和定理 4 以及前面的讨论，我们可以得出以下推论。

推论 1。 设 $\varepsilon < \varepsilon'$。

1. 端点处于 ε'-一致关系的正符号边数量，大于或等于端点处于 ε-一致关系的正边数量。换言之，一致关系具有单调性。

2. ε'-轻顶点的数量可能多于、少于，甚至等于 ε-轻顶点的数量。换言之，轻重关系不一定具有单调性。

其思路是针对每个新的 ε 值重新计算 NonAgreement，所需时间为 $\mathcal{O}(m \times \alpha(G))$；如引理 2 的证明所述，在时间 $\mathcal{O}(n)$ 内判定 G 中顶点的 ε-重度，并在时间 $\mathcal{O}(m+n)$ 内计算图 G^+ 的连通分量作为输出。总的来说， Baseline 的时间复杂度为 $\mathcal{O}(m \times (2+\alpha(G)) + n)$。

总之，使用索引结构，我们投入 $\mathcal{O}(m)$ 的空间（引理 2）和 $\mathcal{O}(m \times (\alpha(G) + \lg m))$ 的时间（引理 3）来构建 $\mathbb{NAO}(G)$，从而能够在时间 $\mathcal{O}(m(\alpha(n)+1))$ 内回答每个具有不同 ε 值的查询（定理 2）。将其与该方法所需的 Baseline $\mathcal{O}(m \times (2+\alpha(G)) + n)$ 时间进行比较可知，对于 ε 的不同取值，我们的基于索引的结构能够加快查询速度。

给定对 NAO 的访问权限，算法 2 构造图 $\widetilde{G^+}$。注意，算法 1 和算法 2 的输出相同。由于判断顶点 v 是否为 ε-重顶点，或判断边 $\{u,v\}$ 的端点是否属于 ε-一致，可在 $\mathcal{O}(1)$ 时间内完成，因此算法 2 的运行时间为 $\mathcal{O}(\max\{|V|, |E^+|\})$，用于执行 **for** 循环，以及 $\mathcal{O}\left(|V_{\widetilde{G^+}}| + |E^+_{\widetilde{G^+}}|\right)$，用于计算 $\widetilde{G^+}$ 的连通分量。总的来说，ICC 中查询算法的运行时间为 $\mathcal{O}(m+n)$，而 CC 的运行时间为 $\mathcal{O}(m \times (2+\alpha(G)) + n)$。

为了对图在 ε 的不同取值下进行查询建模，我们使用一个定义为 ε-Schedule 的

$$\varepsilon\text{-Schedule} = \{\varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_\ell\} \subseteq [0,2]. \tag{6}$$

注意，为了简化记号，我们将 ε-Schedule 视为一个严格递增的实数集合。在实际场景中，任何时候都可以针对 ε 的任意取值请求一次聚类。

算法 2 采用 $\mathbb{NAO}(G)$ 结构的基于索引的相关聚类算法。

1：过程 IndexCorrelationClustering**(**G**,** $\mathbb{NAO}$**(**G**)** ε**) 2：对于所有** $v \in V$ **执行 3：使用** $\mathbb{NAO}$**(()** v**) 识别并移除所有属于** ε-NonAgreement **的边** $\{v,u\}$ **4：使用** $\mathbb{NAO}$**(**v**) 识别并移除满足** u **和** v **均为** ε**-轻量的边** $\{u,v\}$ **5：结束循环 6：将剩余图的连通分量作为聚类结果返回。7：结束过程**

3.2 维护索引

第 3.1 节中介绍的索引结构用于针对用户定义的 ε 动态值计算静态图的聚类。本节将修改该结构，使其适用于针对 ε 的动态值计算动态图的聚类。

CorrelationClustering 的底层图有三种不同的适用操作：(1) 翻转边 $e = \{u,v\}$ 的符号，(2) 添加新顶点 v，以及 (3) 移除现有顶点 v。引理 6、7 和 8 分别讨论了这些操作。

Shakiba 在 [12] 中证明，翻转边 $\{u,v\}$的符号后，两个端点均不在其端点的正邻域并集中的边的 ε-一致性不会改变（ [12]中的命题 2 和 4）。因此，我们只需计算正边 $\{x,w\}$ 的 ε-一致性，其中 $x,w \in (N_{G^+}(u) \cup N_{G^+}(v))$。

引理 6. 假设边 $e = \{u,v\}$ 的符号被翻转。

1. 假设翻转前边的符号为 $+$，那么由于该边现在带有负符号，u 和 v 将不再 ε一致。顶点 v 和 u 分别从数组 $\mathbb{NAO}(u)$ 和 $\mathbb{NAO}(v)$ 中移除。此外，对于所有顶点 $w \in \mathbb{NAO}(u) \cup \mathbb{NAO}(v)$，我们需要更新 $\mathbb{NAO}(w)$ 中顶点 u 和 v 的非一致性值。

2. 假设翻转前边的符号为 $-$，那么将计算它们的 $\text{NonAgreement}_{G^+}(u,v)$，并将顶点 v 和 u 分别添加到 $\mathbb{NAO}(u)$ 和 $\mathbb{NAO}(v)$ 中适当的排序位置。此外，我们需要为所有 x、$w \in \mathbb{NAO}(u) \cup \mathbb{NAO}(v)$ 重新计算 $\text{NonAgreement}_{G^+}(x,w)$，并更新它们的 ε- 一致性值。

应用这些更改的时间复杂度为 $\mathcal{O}\left(\sum_{\{x,w\}\in E^+\cap X\times X}(\log\deg_{G^+}(x)+\log\deg_{G^+}(w))\right)$，其中 $X=\mathbb{NAO}(u)\cup\mathbb{NAO}(v)$。

证明。该引理的正确性可由前面的讨论得出。因此，我们只给出时间复杂度分析。在第一种情况下，找出每个 $\mathbb{NAO}$ 中的顶点需要 $\mathcal{O}(\log \deg_{G^+}(u)+\log \deg_{G^+}(v))$。移除操作的摊销时间为 $\mathcal{O}(1)$。

第二种情况，即将符号从 $-$ 翻转为 $+$，代价更高，因为这可能需要更改集合 X 中两个端点都在其中的所有正边。在最坏情况下，可以假设，两个端点都位于 X 内的边之间的所有 ε-一致性都发生了变化。因此，我们需要更新它们各自对应的索引。根据 $\text{NonAgreement}_{G^+}(u,v)$，我们需要分别将顶点 v 和 u 添加到 $\mathbb{NAO}(u)$ 和 $\mathbb{NAO}(v)$ 中。在排序后的 $\mathbb{NAO}$s 中找到它们的正确位置，可通过 $\mathcal{O}(\log \deg_{G^+}(u)+\log \deg_{G^+}(v))$ 次比较完成。同样，将它们添加到对应位置需要 $\mathcal{O}(1)$ 的摊销时间。对于集合 X 中其他两端点均在其中的正边，例如 $\{x,w\}$，我们可能需要更新它们的 ε-一致性。不失一般性，假设 $\deg_{G^+}(x)\le\deg_{G^+}(w)$。这需要重新计算 $\text{NonAgreement}_{G^+}(x,w)$，耗时 $\mathcal{O}(\min\{\deg_{G^+}(x),\deg_{G^+}(w)\})$；在 $\mathbb{NAO}(x)$ 中查询 NonAgreement，耗时 $\mathcal{O}(\log \deg_{G^+}(x))$；然后可能还要调整其位置，在 $\mathbb{NAO}(x)$ 和 $\mathbb{NAO}(w)$ 中均需 $\mathcal{O}(\log \deg_{G^+}(x)+\log \deg_{G^+}(w))$。最坏情况下，可能需要更新 X 中的所有元素。因此，这种情况可以在

$$\mathcal{O}\left(\sum_{\{x,w\}\in E^+\cap X\times X}(\log \deg_{G^+}(x)+\log \deg_{G^+}(w))\right),$$

在最坏情况下。 □

利用引理 6，我们可以给出 $\text{NAO-FlipEdge}(u,v)$ 算法（算法 3），该算法会翻转边 $\{u,v\}$ 的符号。该算法的正确性和运行时间可直接由引理 6 得出。在某些情况下，还会给出一组正边 E_v^+，它们与一个新添加的顶点 v 相关。可以先添加

算法 3 用于在 $\mathbb{NAO}(G)$ 中应用翻转边 (u,v) 符号所产生效果的 $\text{NAO-FlipEdge}(u,v)$ 算法。

1: 过程 $\text{NAO-FlipEdge}(u,v)$ **2: 令** $\mathcal{A}\leftarrow\mathbb{NAO}(u)\cup\mathbb{NAO}(v)$。**3: 如果边** $\{u,v\}$ **的符号为正，则 4: 通过移除边** $\{u,v\}$**来更新图** G^+。**5: 分别从** $\mathbb{NAO}(v)$ **和** $\mathbb{NAO}(u)$ **中移除顶点** u **和** v。**6: 否则▷ 边** $\{u,v\}$ **的符号为负 7: 通过添加边** $\{u,v\}$**来更新图** G^+。**8: 计算** $\text{NonAgreement}_{G^+}(u,v)$。**9: 分别将顶点** u **和** v **添加到** $\mathbb{NAO}(v)$ **和** $\mathbb{NAO}(u)$ **中。10: 结束 if 11: 对所有** $w\in\mathcal{A}$ **执行 12: 重新计算** $\text{NonAgreement}_{G^+}(u,w)$**，并更新** $\mathbb{NAO}(u)$ **和** $\mathbb{NAO}(w)$。**13: 重新计算** $\text{NonAgreement}_{G^+}(v,w)$**，并更新** $\mathbb{NAO}(v)$ **和** $\mathbb{NAO}(w)$。**14: 结束 for 15: 结束过程**

具有所有负边的顶点，然后翻转 E_v^+ 中的所有边，使它们变为正符号。不过，实际应用中，批量操作的性能会更高。

引理 7. 假设将一个顶点 v 添加到图 G 中，并带有新的正号边 E_v^+。那么，

1. 如果 $E_v^+ = \emptyset$，那么 $\mathbb{NAO}(G)$ 无需进行任何更新，只需在 $\mathbb{NAO}(G)$ 中以常数时间添加一个新的 $\mathbb{NAO}(v)$，并将 v 作为其唯一元素。

2. 否则，令 $X = N_{G^+}(v) \cup \left(\cup_{x \in N_{G^+}(v)} N_{G^+}(x)\right)$。将 E_v^+ 中所有新的正边添加后，我们需要为每条满足 x、$w \in X$ 的正边 $\{x, w\}$ 计算 $\textsc{NonAgreement}_{G^+}(x, w)$，并可能更新 $\mathbb{NAO}(x)$ 和 $\mathbb{NAO}(w)$。这需要

$$\mathcal{O}\left(\sum_{\{x,w\} \in (E^+ \cup E_v^+) \cap X \times X} (\log \deg_{G^+}(x) + \log \deg_{G^+}(w))\right), \tag{7}$$

在最坏情况下进行相应次数的操作。

证明。 由引理 6 可立即得出正确性。同样，第一种情况可在 $\mathcal{O}(1)$ 时间内完成。对于第二种情况，我们需要计算 NonAgreement 的 $|X|$ 个值，并将其添加或更新到对应的 $\mathbb{NAO}$ 中。这需要

$$\mathcal{O}\left(\sum_{\{x,w\} \in (E^+ \cup E_v^+) \cap X \times X} (\log \deg_{G^+}(x) + \log \deg_{G^+}(w))\right), \tag{8}$$

由与引理 6 完全相同的论证可得。 □

备注 1。 比较引理 6 和引理 7 所需的时间时，请注意，集合 X 的大小在引理 7 中可能远大于引理 6 中的大小，这取决于新顶点 v 的 E_v^+ 大小。

NAO-AddVertex(x, N_x) 算法（算法 4）将一个新顶点 x 及其所有相邻正号边 N_x 添加到 G^+，并更新 $\mathbb{NAO}(G)$。该算法的正确性和运行时间可直接由引理 7 得出。对于最后一项操作，移除一个已有顶点 v 及其正

算法 4 将一个新顶点 x 及其所有正邻居N_x 添加到 $\mathbb{NAO}(G)$ 的 NAO-AddVertex(x, N_x) 算法。

1: 过程 NAO-AddVertex(x, N_x) **2: 通过添加新顶点** x**来更新图** G^+ **。 3: 构造一个新的** $\mathbb{NAO}(x)$ **并将其添加到** $\mathbb{NAO}(G)$**。 4: 令** $X \leftarrow N_x \cup (\cup_{z \in N_x} N_{G^+}(z))$**。 5: 对所有** $y \in X$ **执行 6: 对所有** $z \in X \setminus y$ **执行 7: 如果** $\textsc{NonAgreement}_{G^+}(y, z)$ **发生变化，则更新** $\mathbb{NAO}(y)$ **和** $\mathbb{NAO}(z)$**。 8: 结束循环 9: 结束循环 10: 结束过程**

相邻边 E_v^+，可以先将 E_v^+ 中与其相邻的所有边的符号从 $+$ 翻转为 $-$，然后从 $\mathbb{NAO}$ 中移除其 $\mathbb{NAO}(v)$。与顶点添加类似，我们也可以批处理模式执行此操作，以期在实践中获得更好的性能。

NAO-RemoveVertex(x) 的算法与 NAO-AddVertex(x, N_x) 算法（算法 4）类似。

引理 8. 假设从图 G 中移除一个现有顶点 v，其相邻正符号边集合为 E_v^+。那么，

1. 如果 $E_v^+ = \emptyset$，那么 $v(v)$ 仅包含它自身这一项，因此可以轻松地从 $\mathbb{NAO}(G)$ 中移除。其他内容均无需更改，因此时间复杂度为常数。

2. 否则，令 $X = N_{G^+}(v) \cup \left(\cup_{x \in N_{G^+}(v)} N_{G^+}(x) \right)$。我们需要在移除 E_v^+ 中的所有边之后，计算每条正边 $\{x, w\}$ 的 $\textsc{NonAgreement}_{G^+}(x, w)$（其中 $x, w \in X$），并可能更新 $\mathbb{NAO}(x)$ 和 $\mathbb{NAO}(w)$。在最坏情况下，这需要

$$\mathcal{O}\left(\sum_{\{x,w\} \in (E^+ \cup E_v^+) \cap X \times X} (\log \deg_{G^+}(x) + \log \deg_{G^+}(w)) \right), \tag{9}$$

次操作。

证明。 该引理的正确性可由引理 6 推得。运行时间的讨论与引理 7 的证明完全相同。

□

4 实验

为了评估所提方法，我们使用了 7 个由用户间交互构成的图。这些数据集在表 1 中进行了描述，并可通过 SNAP [8] 获取。最小的数据集包含 22 470 个节点和 171 002 条边，最大的数据集包含 317 080 个节点和 1 049 866 条边。在所有这些图中，我们将现有边视为正符号，将不存在的边视为负符号。

每个数据集中 $\textsc{NonAgreement}$s 的分布如图2a至图8a所示。几乎所有数据集中，边的 $\textsc{NonAgreement}$s 分布都服从正态分布，只有 Arxiv ASTRO-PH（图2a）、Arxiv COND-MAT（图4a）和 DBLP（图 8a）存在少许偏差。此外，表2给出了这些分布的更详细统计信息。一个值得注意的现象是，在所有样本数据集中， $\textsc{NonAgreement}$ 最常见的取值为 1。为什么会这样？不失一般性，假设 $\deg_G(u) \geq \deg_G(v)$。那么， $|N_G(v)| = 2\,|N_G(u) \cap N_G(v)|$。根据假设，$|N_G(u)| \geq 2\,|N_G(u) \cap N_G(v)|$，也就是说，顶点 u 和 v 的邻域交集至多包含顶点 u 邻域的一半。此外，顶点 v 的邻域中恰好有一半落在顶点 u 和 v 的邻域交集中。直观地说，顶点 u 对额外顶点的聚类偏好，至少与顶点 u 和 v 都有聚类偏好的顶点数量相同。类似地，顶点 v 对额外顶点的聚类偏好，恰好是顶点 u 和 v 都有聚类偏好的顶点数量的一半。

对于每个数据集，我们根据其$\textsc{NonAgreement}$的分布，使用了不同的一组 ε-$\textsc{Schedule}$s。具体而言：(1) 我们将每个数据集中非一致度的值按非递减顺序排序，并保留重复值。(2) 然后，从这些值中等间距地选取21个不同的值。(3) 将 ε-$\textsc{Schedule}$ 设为第二步选取的值；如果开头没有0，则在开头添加0；如果末尾没有1.99，则在末尾添加1.99。每个数据集的ε-$\textsc{Schedule}$ 总数为21、22或23。

一个有趣的观察结果是图2b至图8b中 $\varepsilon \approx 1^-$ 的簇数量。请注意，我们用 1^-表示区间 $[1 - \epsilon, 1]$，其中 $\epsilon > 0$是某个非零常数。当 $\varepsilon \approx 1$ 但小于1时，簇数量最少。当它逐渐接近1时，簇数量会增加；与它从0接近1时簇数量的减少相比，增加幅度大得多。

如推论 1 所述，随着 ε 的增加， ε-一致顶点的数量是一个非递减函数；图2a至图8a中的绘图证实了这一点，因为 ε-非一致顶点的数量是一个非递增函数。通过更仔细地观察这些图，我们可以看到，在所有这些图中， ε-非一致顶点数量的绘图形状几乎相同，其拐点约在 $\varepsilon \approx 1$ 的值处。这是由于 $\textsc{NonAgreement}_G(u, v)$ 的内在性质所致。

同样，在相同图中观察到推论 1 所述的非单调性结果，涉及 ε-轻顶点的数量。通过目视检查发现，除 Arxiv HEP-TH（图 5a）和 EU-Email（图 7a）外，几乎所有数据集中 ε-轻顶点的数量趋势都相同：随着 ε 增加， ε-轻顶点的数量先增加至某个点（第一区间），然后略微下降（第二区间），并且

最后再次增加，并渐近地趋近于图中顶点的数量（最后一个区间）。对于 ArxivHEP-TH 和 EU-Email，趋势相同，但第二区间非常短。

表 1：数据集描述。

数据集	节点	边
Arxiv ASTRO-PH [7]	18 772	198 110
MUSAE-Facebook [11]	22 470	171 002
Arxiv COND-MAT [7]	23 133	93 497
Arxiv HEP-TH [6]	27 770	352 807
Enron-Email [9]	36 692	183 831
EU-Email [7]	265 214	420 045
DBLP [13]	317 080	1 049 866

表2：每个数据集中 NonAgreement 的统计信息。

数据集	不同的	最小值	最大值	最常见的前 2 个值	
Arxiv ASTRO-PH	16 436	0.015 873 0	1.967 21	1 — 9 664	0.5 — 2 992
MUSAE-Facebook	12 988	0.031 746	1.978 02	1 — 18 534	1.25 — 2 346
Arxiv COND-MAT	3 893	0.044 444 4	1.964 91	1 — 12 018	0.5 — 4 954
Arxiv HEP-TH	23 285	0.086 956 5	1.976 19	1 — 24 852	1.333 33 — 4 910
Enron-Email	20 273	0.090 909 1	1.954 55	1 — 31 704	0.5 — 6 796
EU-Email	28 612	0.117 647	1.990 52	1 — 441 520	0.997 658 — 682
DBLP	11 611	0.013 793 1	1.981 13	1 — 167 590	0.5 — 67 238

朴素相关聚类算法和所提出的基于索引的相关聚类算法均使用 C++[1] 实现，且未进行任何并行化；实验在 Ubuntu 22.04.1 TLS 系统上进行，使用配备 12 GB 随机存取存储器的英特尔 Core i7-10510U 处理器（@ 1.80GHz）。图 2d 至图 8d 给出了运行朴素相关聚类算法（算法 1，本文记为 CC）以及基于索引的相关聚类算法（记为 ICC）所需的时间。请注意，这些图中报告的 ICC 时间不包括构建所需的 $\mathbb{NAO}$的时间，因为这些 $\mathbb{NAO}$只需构建一次，并可在整个 ε-Schedule过程中使用。读取图以及构建 $\mathbb{NAO}$的运行时间以毫秒为单位列于表 3。CC 和 ICC 算法相同，区别在于：对于 CC，每给定一个 ε值，都会计算边的非一致性值以及顶点的 ε-轻度；而对于 ICC，这些值只需在所提出的 $\mathbb{NAO}$结构中计算并存储一次，即可用于针对不同 ε值进行聚类。从图 2d 至图 8d 可以看出，ICC 的运行时间（不包括一次性构建 $\mathbb{NAO}$的时间）远小于 CC。平均而言，我们针对所述 ε-Schedule 的方法使聚类时间减少了 %25。取得这一提升的代价是预先计算 $\mathbb{NAO}$，其耗时平均相当于单次运行 CC 所需时间的 %34；这一开销相当小，因此当需要针对不同 ε值进行多次聚类时，ICC 效率较高。

5 结论

本文提出了一种新颖的索引结构，以降低相关聚类问题近似算法的整体运行时间。该结构可在 $\mathcal{O}(m \times \alpha(G))$ 时间内构建，所需内存为 $\mathcal{O}(m)$。随后，对于 ε 在 $\mathcal{O}(m+n)$ 中的任意取值，我们都可以输出一个相关聚类；相比之下，普通相关聚类算法的时间复杂度为 $\mathcal{O}(m \times (2+\alpha(G)) + n)$。此外，

[1]https://github.com/alishakiba/Correlation-Clustering-Algorithm-for-Dynamic-Complete-Signed-Graphs-An-Index-based-Approach

表 3：为每个数据集构建 $\mathbb{NAO}$ 所需的时间（毫秒）。

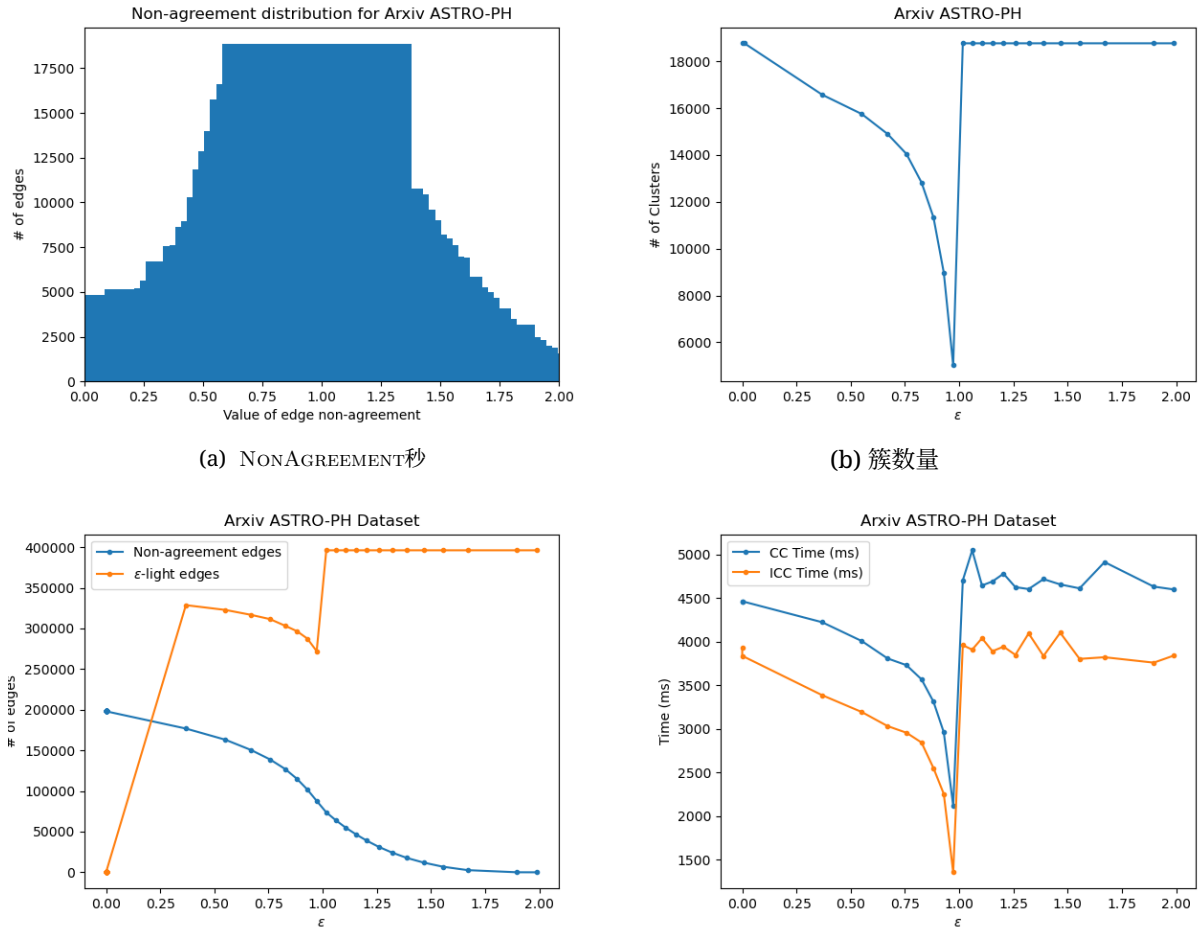
数据集	构建 $\mathbb{NAO}$ 所需的时间（毫秒）	读取图所需的时间（毫秒）
Arxiv ASTRO-PH	1 677	543
MUSAE-Facebook	1 391	427
Arxiv COND-MAT	521	346
Arxiv HEP-TH	12 530	725
Enron-Email	2 498	525
EU-Email	4 479	958
DBLP	4 620	2 167

在底层图发生更新时，包括边符号翻转、顶点添加和顶点删除，所提出的索引都能得到高效维护。理论结果还通过使用七个真实世界图进行的实验得到了实际结果的佐证。实验结果表明，查询的运行时间约减少了 %34。

未来的研究方向之一是研究该算法在 Map-Reduce 等并行框架中的应用，并使其能够扩展到非常大的图。另一个研究方向是提高该算法的近似保证，或从近似比的角度设计更高效的算法。

参考文献

[1] Nikhil Bansal、Avrim Blum 和 Shuchi Chawla。相关聚类。机器学习，56(1):89–113，2004年。[2] Norishige Chiba 和 Takao Nishizeki。森林度与子图枚举算法。*SIAM*计算杂志，14(1):210–223，1985年。[3] Vincent Cohen-Addad、Silvio Lattanzi、Andreas Maggiori 和 Nikos Parotsidis。在线一致相关聚类。载于 Kamalika Chaudhuri、Stefanie Jegelka、Le Song、Csaba Szepesvari、Gang Niu 和 Sivan Sabato 编，第*39*届国际机器学习会议论文集，第162卷，机器学习研究论文集，第4157–4179页。PMLR，2022年7月17日至23日。[4] Vincent Cohen-Addad、Silvio Lattanzi、Slobodan Mitrovi´c、Ashkan Norouzi-Fard、Nikos Parotsidis 和 Jakub Tarnawski。常数轮并行相关聚类。载于国际机器学习会议（*ICML*），第2069–2078页。PMLR，2021年。[5] Thomas H Cormen、Charles E Leiserson、Ronald L Rivest 和 Clifford Stein。算法导论。麻省理工学院出版社，2022年。[6] Jure Leskovec、Jon Kleinberg 和 Christos Faloutsos。随时间演变的图：致密化规律、直径缩小及其可能的解释。载于第*11*届 *ACM SIGKDD* 数据挖掘中的知识发现国际会议论文集，第177–187页，2005年。[7] Jure Leskovec、Jon Kleinberg 和 Christos Faloutsos。图演化：致密化与直径缩小。*ACM*数据知识发现汇刊（*TKDD*），1(1):2–es，2007年。[8] Jure Leskovec 和 Andrej Krevl。SNAP数据集：斯坦福大型网络数据集集合。http: //snap.stanford.edu/data，2014年6月。[9] Jure Leskovec、Kevin J Lang、Anirban Dasgupta 和 Michael W Mahoney。大型网络中的社区结构：自然簇大小与缺乏大型定义明确的簇。《互联网数学》，6(1):29–123，2009年。



(a) NonAgreement秒

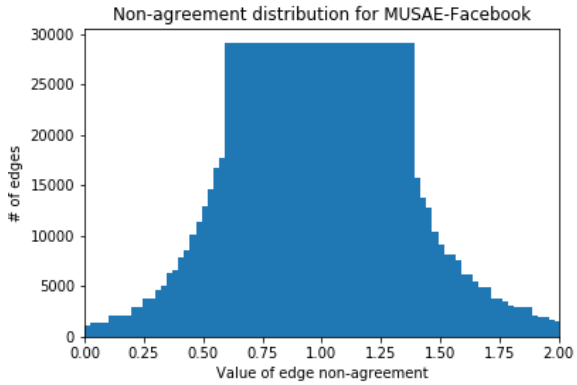
(b) 簇数量

(c) NonAgreement 和 ε-轻边的数量

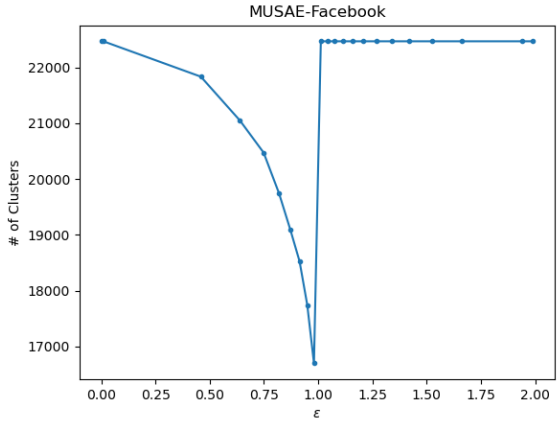
(d) 聚类时间（毫秒）

图2：图 Arxiv ASTRO-PH。

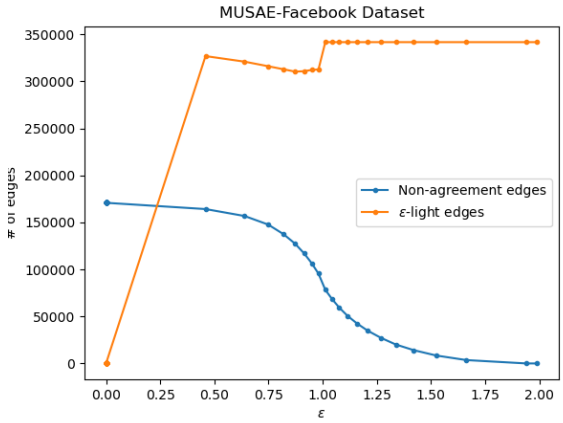
[10] Claire Mathieu、Ocan Sankur 和 Warren Schudy。在线相关聚类。载于第*27*届国际计算机科学理论方面研讨会（*STACS 2010*），第573–584页，2010年。[11] Benedek Rozemberczki、Carl Allen 和 Rik Sarkar。多尺度属性节点嵌入，2019年。[12]Ali Shakiba。动态完全符号图的在线相关聚类。*arXiv*预印本 *arXiv:2211.07000*，2022年。[13] Jaewon Yang 和 Jure Leskovec。基于真实标注定义和评估网络社区。载于*ACM SIGKDD* 数据语义挖掘研讨会论文集，第1–8页，2012年。



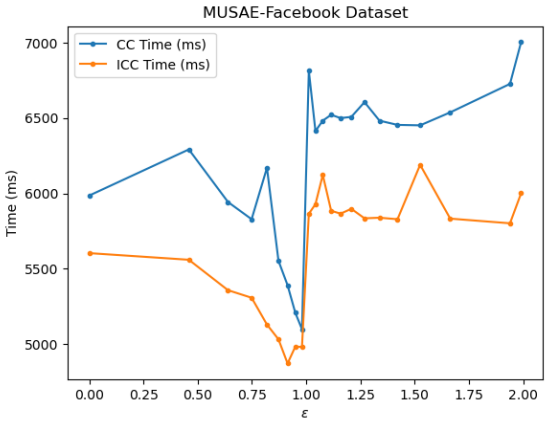
(a) NonAgreement秒



(b) 簇数量

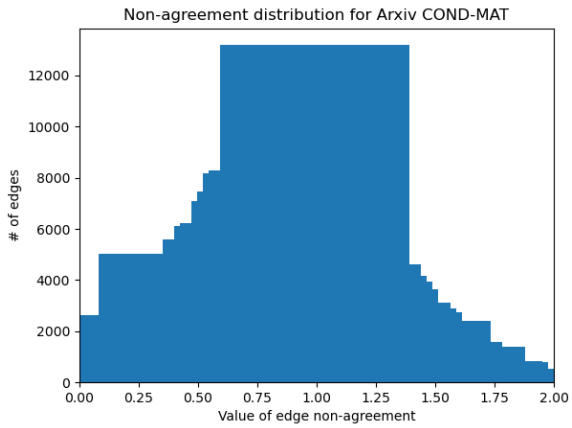


(c) NonAgreement 与 ε-轻边的数量

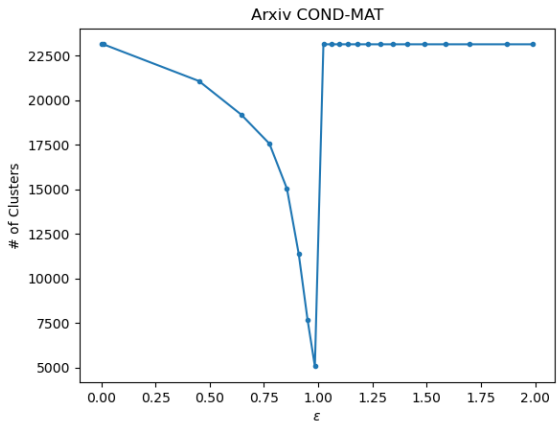


(d) 聚类时间（毫秒）

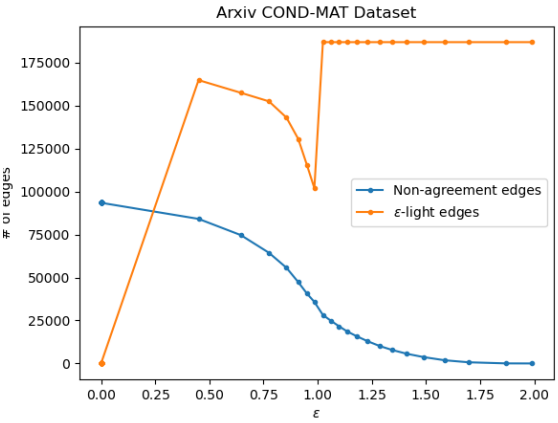
图3：图 MUSAE-Facebook。



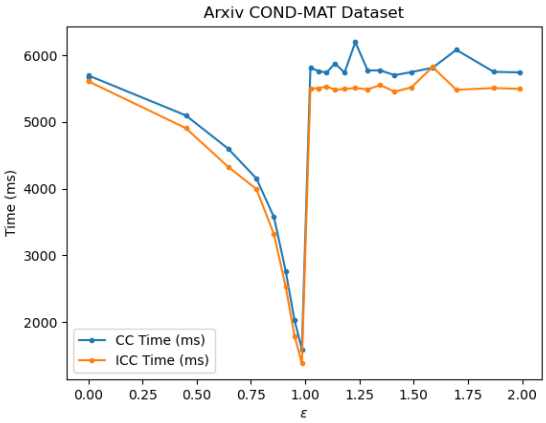
(a) NonAgreements



(b) 簇数量

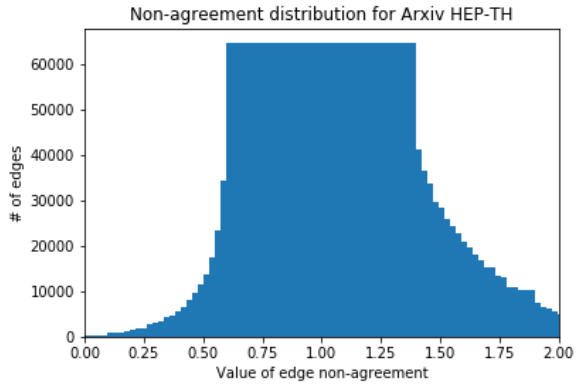


(c) NonAgreement 和 ε-轻边的数量

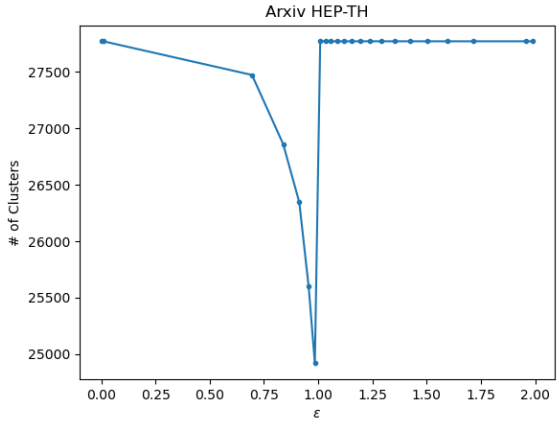


(d) 聚类时间（毫秒）

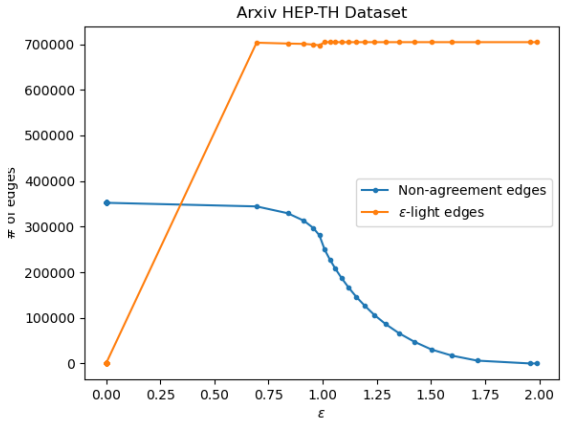
图4：图 Arxiv COND-MAT。



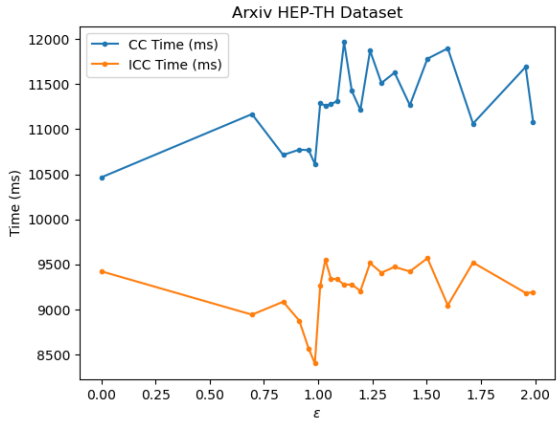
(a) NonAgreement秒



(b) 簇数量

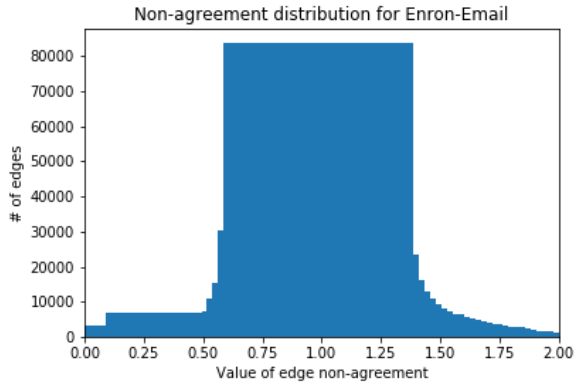


(c) NonAgreement 和 ε-轻边的数量

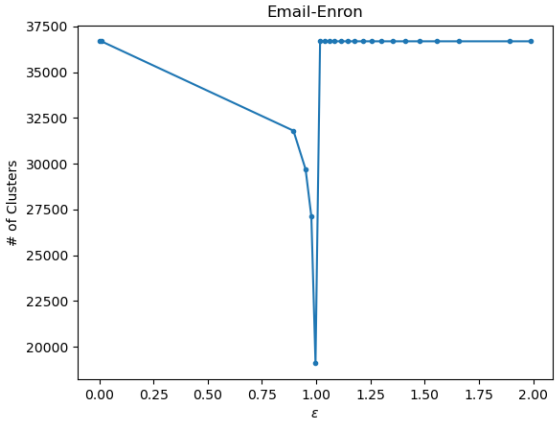


(d) 聚类时间（毫秒）

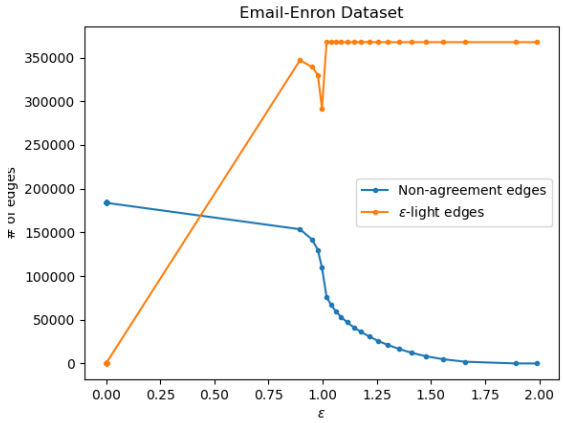
图 5：Arxiv HEP-TH 图。



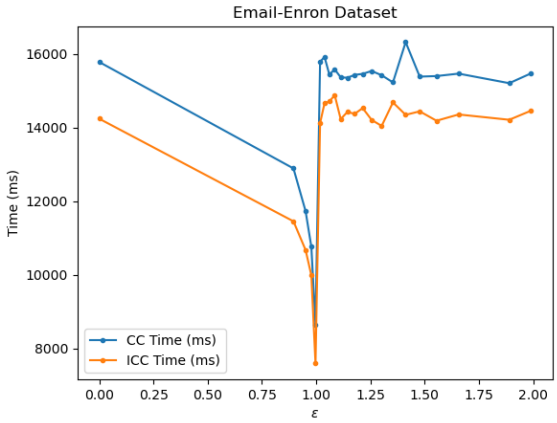
(a) NonAgreement秒



(b) 簇数量

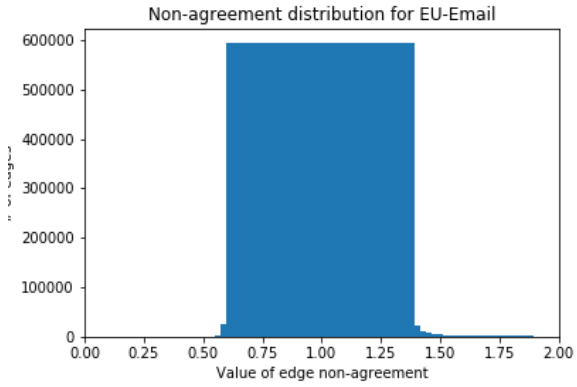


(c) NonAgreement 和 ε-轻边的数量

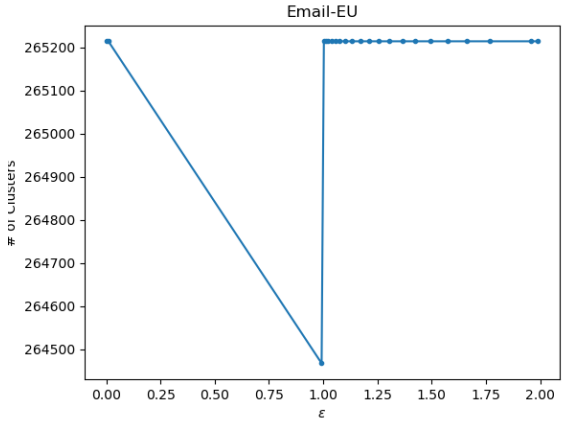


(d) 聚类时间（毫秒）

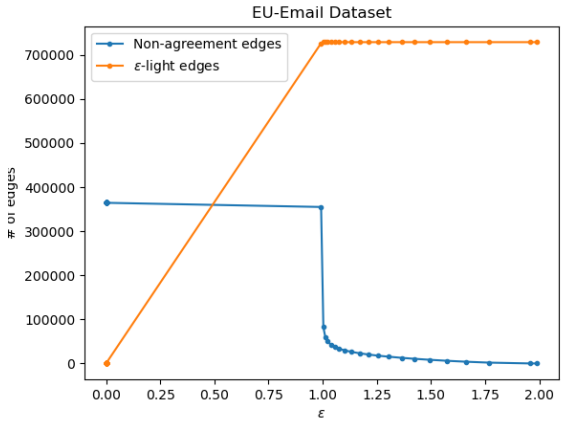
图 6：Enron-Email 图。



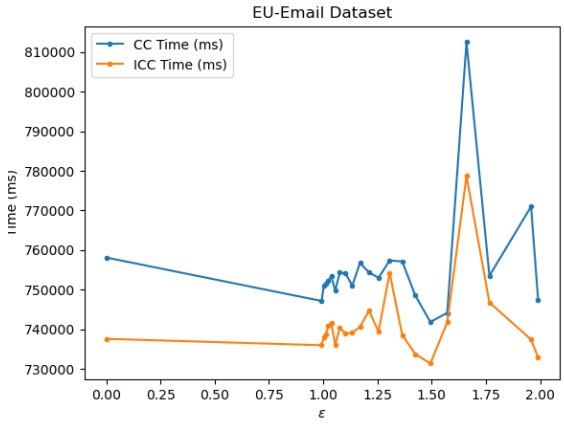
(a) NonAgreements



(b) 簇数量

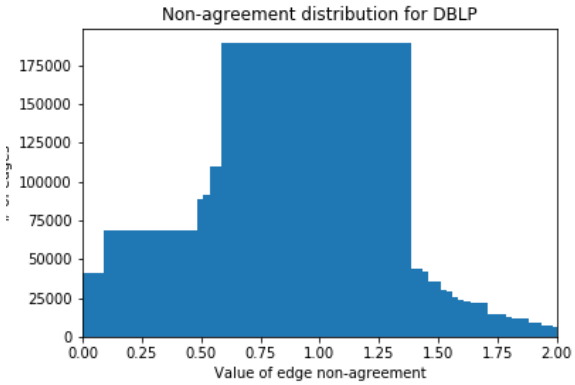


(c) NonAgreement 和 ε-轻边的数量

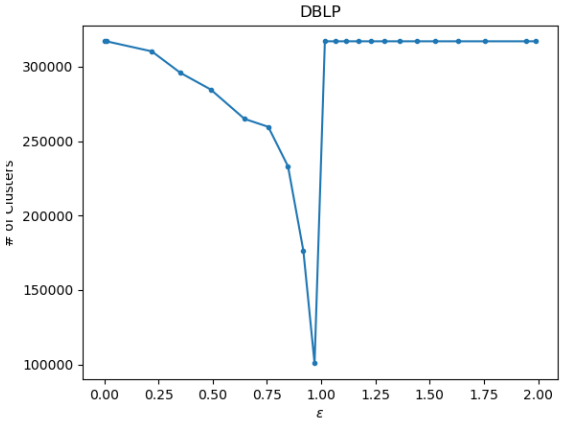


(d) 聚类时间（毫秒）

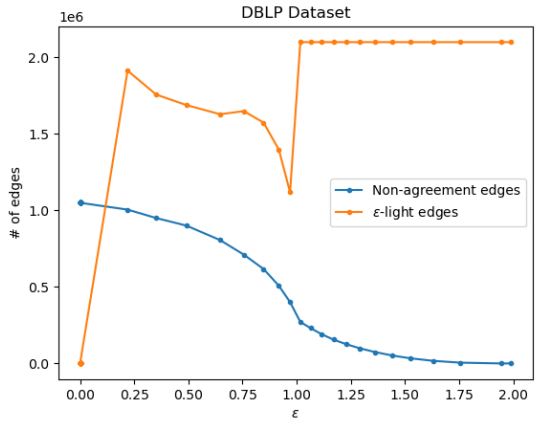
图 7：图 EU-Email。



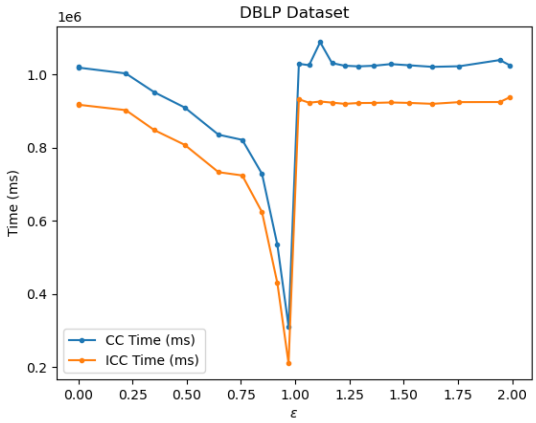
(a) NonAgreement秒



(b) 簇数量



(c) NonAgreement 和 ε-轻边的数量



(d) 以毫秒为单位的聚类时间

图 8：DBLP 图。