# 多级过滤与分区均衡的图相似搜索研究

赵 旭[1,2],梁 平[1,2],顾进广[1,2],高 峰[1,2]
(1. 武汉科技大学 计算机科学与技术学院,湖北 武汉 430065;
2. 智能信息处理与实时工业系统湖北省重点实验室,湖北 武汉 430065)

**摘 要**:图相似搜索指根据图编辑距离,从图数据库中搜索与查询图满足阈值要求的数据图集合的过程。由于图编辑距离的计算是 NP-hard 问题,因此目前主流的图相似搜索算法主要采用“过滤-验证”框架。而其中的过滤算法也仍然是基于分区索引的思想,存在分区过滤过程中过滤下界不紧密、候选集规模大、过滤耗时长和分区大小不均衡等问题。因此提出了对图相似搜索算法相应的改进策略。首先,在整个过滤阶段采用多级过滤的方式,按照代价由低到高的顺序,依次采用图大小、标签,顶点和邻边标签组以及分区索引进行逐级过滤,逐步筛选掉不符合阈值要求的数据图,以减少候选集生成开销;其次,给出了一种启发式方法来指导分区的初始顶点选择,以确保最终的分区尽量均衡,提高分区过滤下界的紧密性。实验结果表明,提出的改进策略不仅可以提高过滤下界的紧密性、降低分区索引构建代价及减少候选图的生成数量,而且缩短了过滤阶段时间及图相似搜索的总体时间。

**关键词**:图相似搜索;图编辑距离;多级过滤;分区索引;分区均衡性
**中图分类号**:TP311 **文献标识码**:A **文章编号**:1673-629X(2025)07-0024-08
doi:10.20165/j.cnki.ISSN1673-629X.2025.0073

# Research on Graph Similarity Search with Multi-level Filter and Partition Balance

ZHAO Xu[1,2], LIANG Ping[1,2], GU Jin-guang[1,2], GAO Feng[1,2]
(1. School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China;
2. Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430065, China)

**Abstract**: Graph similarity search refers to the process of searching for a set of data graphs from a graph database that meet the threshold requirements with respect to the query graph based on the graph edit distance. Since the calculation of graph edit distance is an NP-hard problem, the current mainstream algorithms mainly adopt the "filtering-verification" framework. The filtering algorithm is still based on the idea of partition index, which presents issues such as loose filtering lower bound, large candidate set size, lengthy filtering times, and unbalanced partition sizes during the partition filtering process. Therefore, corresponding improvement strategies for graph similarity search algorithms are proposed. Firstly, in the whole filtering stage, a multi-level filtering approach is employed. In the order of increasing cost, graph size, labels, vertex and adjacent edge label groups, and partition index are used for step-by-step filtering successively, and data graphs that do not meet the threshold requirements are gradually filtered out to reduce the candidate set generation overhead. Secondly, a heuristic method is provided to guide the initial vertex selection for partitioning, ensuring that the final partitions are as balanced as possible and improve the tightness of the partition filtering lower bound. Experimental results show that the proposed improvement strategies not only improve the tightness of the filtering lower bound, reduce the cost of partition index construction and the number of generated candidate graphs but also shorten the time of filtering stage and the overall time of graph similarity search.

**Key words**: graph similarity search; graph edit distance; multi-level filter; partition index; partition balance

**收稿日期**:2024-12-04 **修回日期**:2025-04-08
**基金项目**:国家重点研发计划重点专项(2022YFC3300801);武汉市知识创新专项-曙光计划项目(2023010201020409)
**作者简介**:赵 旭(2000-),男,硕士研究生,CCF 会员(Y1031G),研究方向为图相似搜索和子图同构;通讯作者:梁 平(1975-),女,讲师,博士,研究方向为数据库技术、数据库恢复和数据挖掘;顾进广(1974-),男,教授,博士,CCF 杰出会员(05460D),研究方向为语义网与知识图谱。

## 0　引　言

近两年来,随着 AI 技术的蓬勃发展,许多研究机构和企业为满足模型的训练需要,都迫切需要海量的数据[1]。在海量的数据中,大多数的数据以半结构化或者非结构化的形式展现[2],如何存储和搜索[3]这些数据至关重要。图数据库以原生图模型的方式存储数据,图的顶点表示实体,边表示实体之间的联系,顶点和边的标签携带特征或数据[4],适用于非结构化的数据,可以高效地处理实体之间的复杂关系,提供快速、直接的图搜索操作。因此,其已成功应用于多个领域,如药物研发、社交网络、模式识别和生物信息学等[5-8]。

图相似搜索是依据图编辑距离,从图数据库中搜索与查询图满足阈值要求的数据图集合。由于图编辑距离是 NP-hard 问题,因此大多研究都采用"过滤-验证"框架[9]。过滤阶段通常是创建索引来过滤掉尽可能多的假阳性图,生成候选集;验证阶段则是对候选集逐一进行代价昂贵的 GED 计算。所以,图相似搜索的性能主要与过滤成本、候选集规模和验证代价有关。

在验证阶段,常见的算法有:A$^*$-GED[10]、CSI-GED[11]、BSS_GED[12]、AStar-LSa[13]等。

在过滤阶段,目前较先进的过滤方法是基于分区索引的思想。Zhao 等人提出基于分区的 Pars[14]过滤算法,其主要思想是当阈值为 $\tau$ 时,将数据图划分为 $\tau+1$ 个大小不固定且不重叠的分区,根据鸽巢原理,如果查询图 $q$ 不包含 $\tau+1$ 个分区中的任意一个,则编辑距离一定超过阈值,此时可以将数据图过滤掉。Liang 等人提出 ML-Index[15]算法。该算法将 $\tau+1$ 个分区拓展为带参数 $k$ 的 $\tau+k$ 个分区,并且考虑到分区的大小和标签频率等因素,将随机划分优化为选择性图划分,通过建立多层的倒排索引,可以获得更好的过滤效果。邱等人提出 Z-Index[16]算法,该算法基于扩展概率对查询图分区,并通过构建"零"索引结构,降低索引空间开销。

然而,上述分区方法存在的缺点:(1)直接对所有数据图进行分区索引构建,导致索引构建代价很高,过滤阶段耗时过长;(2)在分区索引构建中,数据图分区的初始顶点采用随机选择的方式,容易导致最终分区的大小不均衡,影响过滤效果,造成候选集规模过大。

针对以上缺点,该文提出了相应的改进策略:

(1)分区过滤采用了多级过滤的方法。在整个过滤阶段,按照过滤代价由低到高,依次采用图大小、标签,顶点和邻边标签组以及分区索引进行逐级过滤,减少候选集生成的时间开销。

(2)提出了一种启发式方法来指导分区的 $\tau+k$ 个初始顶点选择,保证每个顶点都有充足的邻居扩充分区,既平衡了分区的大小、增加了分区过滤下界的紧密性,又减小了候选集的规模。

## 1　问题定义

该文讨论的图为有标签和无向的简单图。图 $g$ 可以表示为一个四元组 $(V(g),E(g),\Sigma,l)$,其中 $V(g)$ 是顶点集合,$E(g)\subseteq V(g)\times V(g)$ 是边集合,$\Sigma$ 是标签集,$l:V(g)\cup E(g)\rightarrow\Sigma$ 是标签函数,它为每个顶点或边分配一个标签。图 $g$ 中顶点和边的数量分别用 $|V(g)|$ 和 $|E(g)|$ 表示。

图编辑操作是指对图进行变换的一系列操作,具体包括以下六种:插入孤立的顶点、删除孤立的顶点、修改顶点的标签、在两个顶点之间插入边、删除两个顶点之间的边、修改边的标签。

定义 1(图编辑距离):将图 $q$ 转换为图 $g$ 所需的最小编辑操作数量,简称为 GED(Graph Edit Distance)。它是两个图相似的衡量标准,图编辑距离越小,两个图越相似。

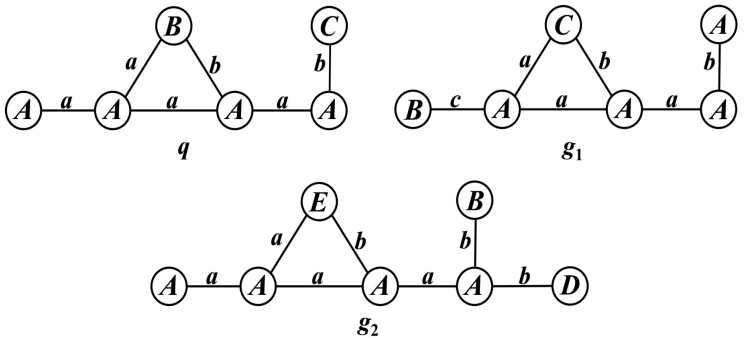


图 1　查询图和数据图示例

例 1:考虑图 1 中的查询图 $q$ 和数据图 $g_2$,将 $q$ 转换为 $g_2$ 的编辑操作步骤具体是:(1)将顶点 $B$ 标签修改为 $E$;(2)将顶点 $C$ 标签修改为 $B$;(3)插入孤立的顶点 $D$;(4)插入连接顶点 $A$、$D$ 的边,因此 GED($q$,$g_2$)=4。

定义 2(图相似搜索):给定查询图 $q$,数据图集合 $G=\{g_1,g_2,\cdots,g_{|G|}\}$ 以及编辑距离阈值 $\tau$,图相似搜索是从 $G$ 中搜索满足 GED($q$,$g$) $\leqslant\tau$ 的数据图集合

$R$,即 $R=\{g \mid GED(q,g) \leqslant \tau\}$。

例 2:在图 1 中,若编辑距离阈值 $\tau=3$,考虑查询图 $q$ 和数据图集合 $G=\{g_1,g_2\}$,因为 $GED(q,g_1)=3 \leqslant \tau$,$GED(q,g_2)=4>\tau$,所以满足编辑距离阈值的数据图为 $g_1$,即 $R=\{g_1\}$。

定义 3(过滤下界):对于图 $q$ 和 $g$,过滤下界是对二者图编辑距离的最小值估计,表示为 $LB(q,g)$。

若 $\tau < LB(q,g) \leqslant GED(q,g)$,说明过滤下界大于编辑距离阈值 $\tau$,则可以安全将图 $g$ 过滤掉,而不用进入代价昂贵的验证阶段。过滤算法的过滤下界越紧密,其生成的候选集越小,过滤性能越好。

## 2 分区过滤算法设计与分析

设计的分区过滤算法流程包括:图大小过滤、标签过滤、VELG 过滤、分区索引过滤、候选集验证。该文主要关注于前三部分的过滤阶段,其中分区索引过滤和验证阶段算法不再赘述,只针对分区索引过滤中分区初始顶点随机选择的缺陷提出改进。设计的分区过滤算法如算法 1 所示。

算法 1:分区过滤算法
输入:数据图集合 $G$,查询图 $q$,图编辑距离阈值 $\tau$
输出:候选图集合 $C$
过程:
1. // 初始化
2. $C=\varnothing$, firstC = $\varnothing$, secondC = $\varnothing$, thirdC = $\varnothing$
3. //图大小过滤
4. for $g \in G$
5. if sizeFilter($q,g$) = true then
6. firstC = firstC $\cup \{g\}$
7. // 标签过滤
8. for $g \in$ firstC
9. if labelFilter($q,g$) = true then
10. secondC = secondC $\cup \{g\}$
11. // 顶点和邻边标签组过滤
12. for $g \in$ secondC
13. if velgFilter($q,g$) = true then
14. thirdC = thirdC $\cup \{g\}$
15. // 分区索引过滤
16. for $g \in$ thirdC
17. if partitionFilter($q,g$) = true then
18. $C = C \cup \{g\}$
19. return $C$ // 返回候选集

### 2.1 图大小过滤

利用数据图 $g$ 与查询图 $q$ 之间的顶点和边数量的差异,可以得到图大小过滤下界 $LB_s$,其计算方式为:

$$LB_s(q,g) = \| V_q \| - | V_g \| + \| E_q \| - | E_g \| \quad (1)$$

其中,$|V_g|$ 和 $|E_g|$ 表示图 $g$ 中顶点和边的数量。若 $LB_s > \tau$,则将图 $q$ 转换为图 $g$ 的编辑操作中至少包含 $\tau+1$ 次顶点和边的添加或删除操作,因此 $GED(q,g) > \tau$,此时可以安全地将数据图 $g$ 过滤掉。

例 3:对于图 1 中的 $q$ 和 $g_2$,若 $\tau=1$,则 $LB_s(q,g_2)=|6-7|+|6-7|=2>\tau$,此时可以将数据图 $g_2$ 过滤掉。

图大小过滤,其缺陷在于当图的大小接近或阈值较小时,过滤效果很差。对于图 1 中的 $g_1$,计算出的过滤下界为 0,此时没有过滤效果。但是由于其时间复杂度为常量级,代价非常低,所以将其作为第一级过滤是非常合适的。

### 2.2 标签过滤

标签过滤是利用两个图的顶点和边标签集的差异进行过滤,标签过滤下界表示为 $LB_1$,其计算公式为:

$$LB_1(q,g) = dif(L_V(q),L_V(g)) + dif(L_E(q),L_E(g)) \quad (2)$$

其中,$L_V(g)$ 表示图 $g$ 的顶点标签集,$L_E(g)$ 表示图 $g$ 的边标签集,$dif(A,B)=\max(|A|,|B|)-|A \cap B|$。

例 4:对于图 1 中的图 $q$ 和 $g_2$,若 $\tau=2$,则 $LB_1(q,g_2)=\max(6,7)-5+\max(6,7)-6=3>\tau$,此时可以将数据图 $g_2$ 过滤掉。

对比图大小过滤下界 $LB_s(q,g_2)=2$,标签过滤下界更加接近于 $GED(q,g_2)$,说明其更加紧密,当阈值 $\tau$ 较大时,标签过滤下界可以过滤更多的假阳性图。标签过滤下界的计算涉及交集运算,其过滤代价高于图大小过滤,因此将其作为第二级过滤。

### 2.3 VELG 过滤

在多级过滤的前两级中,图大小过滤在图的顶点和边数量相近时,效果会很差;标签过滤,虽然考虑到顶点和边的标签集差异,但并没有考虑到图的结构差异,过滤效果有限。因此,提出 VELG 过滤作为第三级过滤,进一步提高下界紧密性,缩小候选集规模。

定义 4(VELG):图的顶点标签和其邻边标签集组成的数据结构称之为顶点和邻边标签组,表示为 $VELG(v)=(l(v),L_{AE}(v))$,其中 $l(v)$ 是顶点的标签,$L_{AE}$ 是该顶点的邻边标签集合,图 $g$ 所有顶点的 VELG 集合表示为 $VELGS(g)$。

例 5:图 1 中,查询图 $q$ 的 VELG 集合 $VELGS(q)=\{(A,\{a\}),(A,\{a,a,a\}),(A,\{a,a,b\}),(A,\{a,b\}),(B,\{a,b\}),(C,\{b\})\}$。

定义 5(VELG 编辑距离):对于顶点 $v_1$ 和 $v_2$,定义 $VELG(v_1)$ 和 $VELG(v_2)$ 的编辑距离 vged 计算方式为:

$$vged(v_1,v_2) = T(v_1,v_2) + \frac{1}{2}dif(L_{AE}(v_1),L_{AE}(v_2))$$

$$T(v_1,v_2) = \begin{cases} 1, l(v_1) \neq l(v_2) \\ 0, l(v_1) = l(v_2) \end{cases} \quad (3)$$

例 6:对于图 1 中的查询图 $q$,顶点 $B$ 的 VELG = ( $B$ ,{ $a$ , b}),顶点 $C$ 的 VELG = ( $C$ ,{b}),它们的 vged = 1+1/2(2−1) = 1.5。

给定查询图 $q$ 和数据图 $g$,可以分别得到 $q$ 和 $g$ 的 VELG 集合 VELGS( $q$ )和 VELGS( $g$ )。若两个图相似,则它们的 VELG 集合一定也相似,因此可以通过 VELGS 映射距离来衡量这种相似性。

定义 6(VELGS 映射距离):给定查询图 $q$ 和数据图 $g$,将 VELGS( $q$ )与 VELGS( $g$ )做一一映射,对于 VELG 数量不足的添加(null, $\varnothing$ ),从而得到一组完整的映射对。各映射对的 VELG 编辑距离之和,即为 VELGS 映射距离。

将所有映射情况中的 VELGS 映射距离最小值的上界作为 VELG 过滤下界 $LB_v$, 表示为:

$$LB_v = \lceil \min_{p} \sum_{u_i \in V_q} \text{vged}(u_i, p(u_i)) \rceil \quad (4)$$

问题的关键在于求解 VELGS 映射距离最小值,以图 1 中的图 $q$ 和图 $g_1$ 为例,可以构造它们的映射图,如图 2 所示。

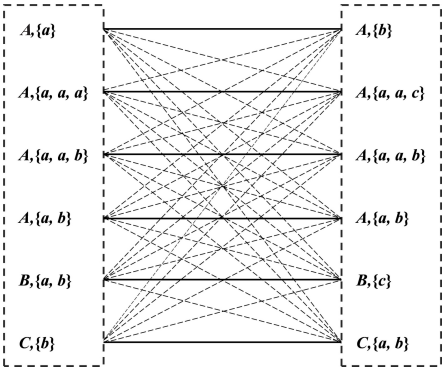


图 2　VELGS 映射

分析可知,该图为典型的二部图,求解最小映射实际上就是求解有权二部图最优匹配问题,其权重是两顶点间的 VELG 编辑距离,因此可以采用经典的 KM 算法进行求解。VELG 过滤的详细流程如算法 2 所示。

算法 2:VELGFilter 算法

输入:查询图 $q$,数据图 $g$,图编辑距离阈值 $\tau$

输出:true or false

过程:

```
1. // 计算 q 和 g 的 VELG 集合
2. compute VELGS( q ), VELGS( g )
3. // 求二者交集
4. I = VELGS( q ) ∩ VELGS( g )
5. //执行 VELG 简化操作
6. VELGS( q ) = VELGS( q ) −I
7. VELGS( g ) = VELGS( g ) −I
8. LB_v = KM − algorithm ( VELGS( q ), VELGS( g ))   // 使用 KM 算法求解下界
9.    if LB_v < τ then
10.      return true
11. else
12.      return false
```

在算法 2 中,首先计算出 VELGS( $q$ )和 VELGS( $g$ ),再进行简化处理,随后使用 KM 算法求解 $LB_v$,最后与阈值 $\tau$ 对比判断是否为候选图。

该算法的性能主要取决于 KM 算法计算 $LB_v$的效率。KM 算法的时间复杂度为 $O(n^3)$ ,其中 $n$ 等于 VELG 集合的大小,这里为了减小 KM 算法的问题规模,采用了先计算 VELGS( $q$ )和 VELGS( $g$ )的交集,再各自减去交集的简化操作。这种简化操作是有效的,因为经过了前两级过滤,此时的候选图与查询图相似度很高,它们的 VELGS 有很大的交集,从而减小了 $n$ 值。

例 7:对于图 1 的查询图 $q$ 和数据图 $g_1$,若 $\tau = 2$,图大小过滤下界 $LB_s = 0$,标签过滤下界 $LB_l = 1$,前两级过滤均不能将该图过滤掉,而 VELG 过滤下界 $LB_v = 3 > \tau$ ,可以将该图过滤掉。实际上,此时计算出的 $LB_v(q, g_1)$与 GED( $q$ , $g_1$)仅相差 1,因此,VELG 过滤下界是非常紧密的。在效率方面,由于做了 VELGS 简化处理,含有 6 个顶点的图,实际计算的 $n$ 值为 4,在更大的图上,这种简化操作效果会更加明显。

### 2.4　分区初始顶点选择

算法 1 在三级过滤后,使用分区索引算法进行最后的过滤,它的过滤下界比前三级的过滤下界更加紧密。其思想是将图数据库中的每个数据图划分为 $\tau + k$ 个半边子图,利用半边子图构造倒排索引,通过验证半边子图是否子图同构于查询图,计数得到每个数据图的子图匹配值 match。若 match 小于 $k$ ,则图编辑距离一定大于 $\tau$ ,可将数据图过滤。

在 ML−Index 算法中已提出较大的分区更容易受到图编辑操作的影响,从而成为一个不匹配的分区,因此构造较大的分区更有利于过滤。然而,图的大小一定,若某些分区过大,其他的分区会过小,而小分区更容易成为匹配分区,因此更好的选择是构造大小均衡的分区。

在数据图划分时,Pars 算法和 ML−Index 算法都采用随机选择的方式确定分区的初始顶点,随后利用初始顶点的邻居顶点来逐步扩展分区。这种方式存在的不足是:随机选择初始顶点不确定性太大,若初始顶点过于集中会导致部分分区没有充足的邻居节点进行扩充,造成最终的分区大小不均衡。

例 8:如图 3 所示,以数据图 $g_1$ 为例,若 $\tau = 1$, $k = 2$,则需要将数据图 $g_1$ 划分为 3 个分区,若分区初始顶点选择 $u_4$、$u_5$、$u_6$,则最终分区大概率为 $p_1$、$p_2$、$p_3$,此

时 $p_2$ 和 $p_3$ 都是查询图 $q$ 的半边子图,满足匹配分区数 match 大于等于 $k$,因此不会将数据图 $g_1$ 过滤。造成这种情况的根本原因是初始顶点选择不合适,导致分区非常不均衡,小分区很容易就成为匹配分区。

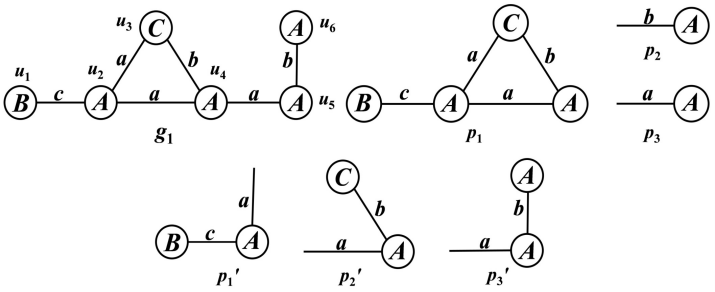


图 3 数据图 $g_1$ 分区示例

这里用 balance( $g$ )来衡量图 $g$ 的分区均匀性,其公式为:

$$\text{balance}(g)=\sqrt{\frac{\sum_{i=1}^{\tau+k}(|V_{p_i}|-|V_g|/(\tau+k))^2}{\tau+k}} \tag{5}$$

其中, $|V_{p_i}|$ 是分区 $p_i$ 的顶点数量, $|V_g|$ 是图 $g$ 的顶点数量。balance 值越小,表示分区越均衡。

由于图分区的复杂性,选取最优的 $\tau+k$ 个初始顶点时间复杂度非常高,并且也不划算,因为这仅仅是图划分的第一阶段。基于以上原因,给出了一种启发式算法——InitialVertexSelection 算法来指导初始顶点选择。

该算法主要考虑两个因素:(1)顶点度数。拥有较大度数的顶点,有充足的邻居用于扩充分区。(2)邻居顶点集。当前选择的顶点不能出现在已选顶点的邻居顶点集中,从而保证初始顶点不会过于集中。InitialVertexSelection 算法如算法 3 所示。

算法 3:InitialVertexSelection 算法
输入:数据图 $g$,参数 $k$,图编辑距离阈值 $\tau$
输出:初始分区的 $\tau+k$ 个顶点序列 InitialV
过程:
1. Seq←DegreeSequence( $g$ )
2. $v$ ←SelectFirstVertex(Seq)
3. InitialV = { $v$ }
4. N(IntialV) = NeighborVertex( $v$ )
5. while InitialV. size < $\tau+k$ do
6. $v$ = SelectNextVertex(Seq)
7. if $v \notin N$(InitialV) then
8. InitialV = InitialV ∪ { $v$ }
9. $N$(IntialV) = $N$(IntialV) ∪ NeighborVertex( $v$ )
10. else
11. if InitialV. size+ $N$(InitialV). size == $|V_g|$ then
12. InitialV = InitialV∪{ $v$ }
13. end while
14. return InitialV

算法 3 首先生成图 $g$ 的顶点序列,其按照度数从大到小的顺序排列,对于度数相同的顶点,则按照顶点的标签频率从低到高排列。然后选择序列的第一个顶点,并初始化 InitialV、$N$(InitialV),其中 $N$(InitialV)表示已选择顶点的邻居顶点集合。依次从序列中选择顶点,如果该顶点不在 $N$(InitialV)中,则将其加入 InitialV,并更新 $N$(InitialV)。如果已无顶点满足 $v \notin N$(InitialV)的条件,此时直接加入顶点到 InitialV 中。

采用算法 3,一种可能的初始顶点选择情况为 $u_4$、$u_1$、$u_6$,后续按照 InitialV 序列的倒序依次去扩展各分区,可能的分区情况如图 3 中 $p_1'$、$p_2'$、$p_3'$ 所示,此时仅有 $p_2'$ 为匹配分区,match 小于 $k$,因此可将数据图 $g_1$ 过滤。

由于数据图进行了预处理,算法 3 的性能主要取决于顶点序列生成,算法时间复杂度为 $O(\log|V_g|)$。

## 3 实验结果与分析

### 3.1 实验环境

实验在 AIDS 数据集上进行,以验证文中算法的性能。AIDS 是来自 NCI/NIH 发展治疗项目的抗病毒筛选化合物数据库,共有 42 687 个图结构化合物,平均有 25.6 个顶点和 27.6 条边,共 62 个顶点标签(如元素 C、N、O 等)和 3 个边标签。

文中算法重点在于过滤阶段,编辑距离阈值设置为 1 到 6。

在实验中,主要考虑以下几个方面的性能评估:(1)过滤性能:多级过滤阶段各级过滤产生的候选图数量以及过滤时间;(2)分区均衡性:主要是 balance( $g$ )的大小,该值越小,分区越均衡;(3)索引构建代价:包括分区索引大小和构建时间;(4)搜索总时间:图相似搜索的总响应时间。

实验选取基于分区思想的 Pars 算法、ML-Index 算法和 Z-Index 算法与文中算法进行对比,在 AIDS 数据集中进行实验验证。为保证实验结果的准确性,每次从数据集中随机选取 1 个数据图作为查询图,并重复实验 100 次取平均值。

实验的运行环境,采用 Windows 11 64 位操作系统,内存为 32 GB,CPU 为 Intel(R) Core(TM) Ultra 5 125H。开发环境为 Microsoft Visual Studio 2022,Pars 算法、ML-Index 算法和文中算法均使用 C++语言实现。

### 3.2　实验结果分析

#### 3.2.1　过滤性能分析

首先评估文中算法中多级过滤阶段各级过滤的性能。如图 4 和图 5 所示,分别是多级过滤阶段各级过滤得到的候选图数量和过滤时间对比。在候选图数量方面,同一阈值下,大小过滤、标签过滤、VELG 过滤和分区索引过滤得到的候选图数量依次显著降低。在过滤时间方面,图大小过滤耗时最短,VELG 过滤时间高于标签过滤约 1 个数量级,而分区索引过滤时间又高于 VELG 过滤约 1 个数量级。以上结果说明了图大小、标签、VELG 以及分区的过滤下界紧密性是逐级递增的,并且过滤代价也是逐级递增的,因此多级过滤顺序是合理的。

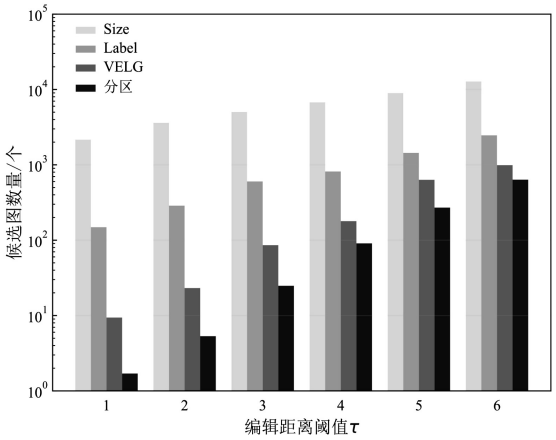


图 4　各级过滤候选图数量对比

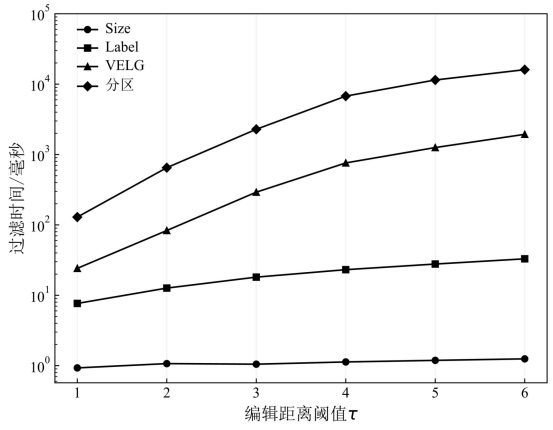


图 5　各级过滤时间对比

图 6 是 Pars、ML-Index、Z-Index 和文中算法过滤时间对比。由于采用了多层索引结构,ML-Index 算法过滤阶段所需时间高于 Pars。文中算法同样采用了多层索引,但其过滤时间最短,约为 Pars 算法的 40% ~80%,ML-Index 算法的 15% ~35%,以及 Z-Index 算法的 60% ~90%。这表明多级过滤逐级缩小候选图数量的机制,可以有效缩短过滤时间。

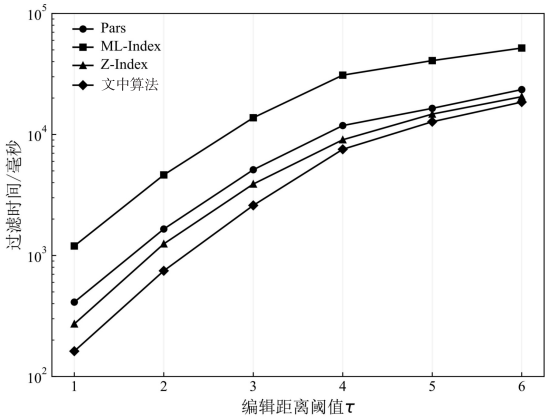


图 6　各算法过滤时间对比

#### 3.2.2　分区均衡性分析

图 7 是 Pars、ML-Index 和文中算法分区均衡性对比结果,其中算法名称中带 initial 表示在原有算法基础上添加分区初始顶点选择算法。文中算法的 balance( $g$ )大于 Pars,而小于 ML-Index 约 16%。大于 Pars 是因为文中算法与 ML-Index 一样采用了选择性分区策略,一定程度上破坏了分区均衡性;小于 ML-Index 是因为文中算法采用了多级过滤,经过前三级的过滤,此时参与分区的候选图大小非常接近。另外,添加分区初始顶点选择算法的 Pars_initial,其 balance( $g$ )小于原有算法 25% ~30%,ML-Index_initial 小于原有算法 6% ~11%,进一步证明了分区初始顶点选择算法可以有效提高分区均衡性。

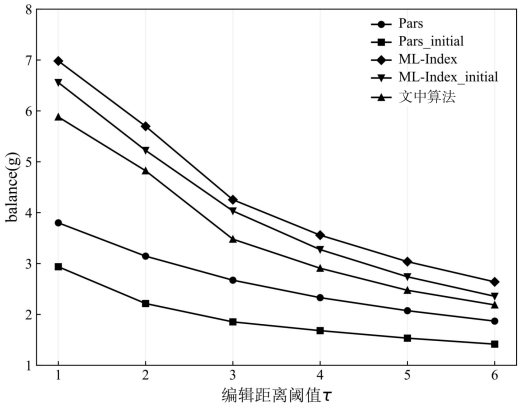


图 7　各算法均衡性对比

图 8 是各算法的候选图数量对比。Pars_initial 算法候选图数量小于原有算法 15% ~25%,ML-Index_initial 算法小于原有算法 10% ~15%,证明分区初始顶点选择算法可以提升过滤下界紧密性,提高过滤效果。文中算法由于采用多级过滤以及分区初始顶点选择算法,其候选图数量小于 Pars 算法 50% ~70%,小于 ML-Index 算法 20% ~30%,小于 Z-Index 算法

5% ~8%,说明文中算法具有更好的过滤效果。

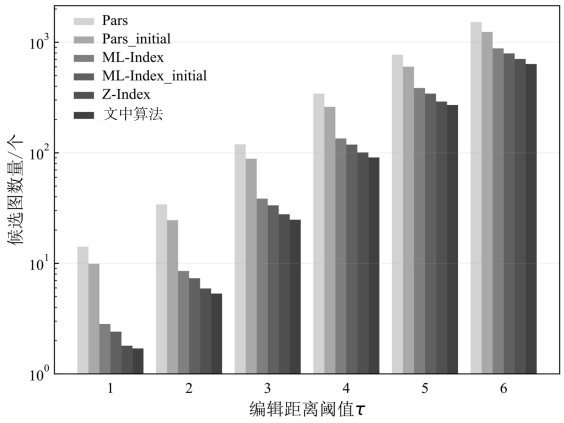


图 8 各算法候选图数量

### 3.2.3 索引构建代价分析

索引构建代价包括分区索引大小和索引构建时间,实验结果分别如图 9 和图 10 所示。

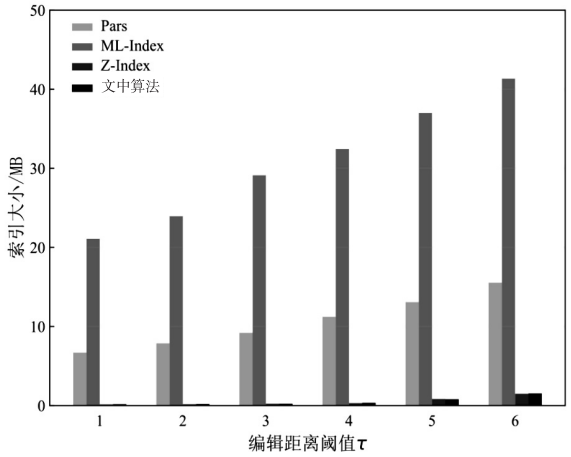


图 9 各算法索引大小对比

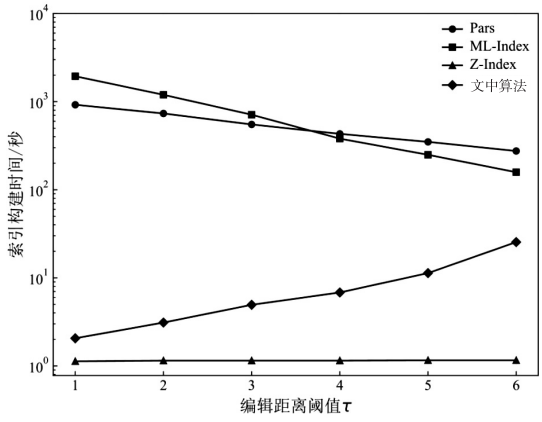


图 10 各算法索引构建时间对比

从图 9 可知,在索引大小上,文中算法远小于 Pars 和 ML-Index 算法,约为 Pars 算法的 5%,ML-Index 算法的 2%,与 Z-Index 算法基本持平。这是因为:一方面,多级过滤使参与分区索引构建的数据图数量大幅降低;另一方面,分区初始顶点选择算法减少了不同分区的数量,进一步降低了索引的大小。由于 Z-Index 只对查询图分区,同时采用了索引压缩策略,所以索引内存占用很小,文中算法的索引大小与之基本持平,进一步说明了在减小索引大小方面的有效性。

从图 10 可知,在索引构建时间上,Pars 和 ML-Index 算法索引构建时间都随阈值增大而缩短,这是因为这两种算法没有多级过滤,所有数据图都参与索引构建,阈值增大,分区变小,分区同构的计算时间得以缩短。而文中算法索引构建时间随阈值增大而增大,这是因为多级过滤后数据图数量大幅降低,此时影响索引构建时间的主要是参与索引构建的数据图数量,其索引构建时间相比 Pars 和 ML-Index 算法下降了约两个数量级。文中算法索引构建时间要长于 Z-Index 算法,这是因为 Z-Index 算法只对查询图分区,其数量一定,因此索引构建时间基本恒定。但是,文中算法在其他性能指标上均优于 Z-Index 算法。综上所述,文中算法在索引大小和构建时间方面都表现出较好的性能,有效降低了索引构建代价。

### 3.2.4 搜索总时间分析

文中算法搜索总时间包括分区索引构建时间、多级过滤产生候选图的时间和图编辑距离验证的时间。如图 11 所示,在不同阈值下,文中算法的搜索总时间均小于 Pars 算法和 ML-Index 算法,对比 Pars 算法降低了约 60%,对比 ML-Index 算法降低了约 30%,对比 Z-Index 算法降低了约 7%。证明了文中算法有效缩短了图相似搜索的总时间。究其原因是因为文中算法过滤阶段耗时最短,生成候选图的数量最少,图编辑距离验证的时间也是最短的,因此总体搜索时间达到最短。

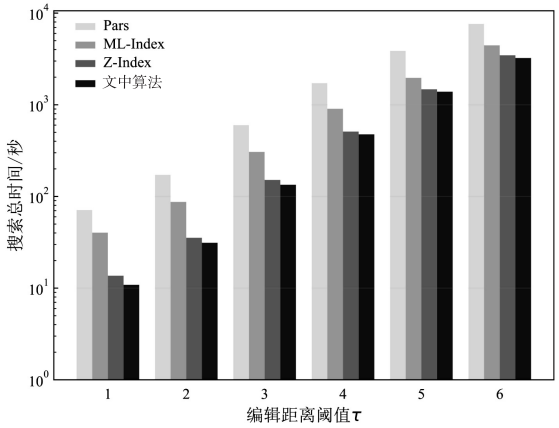


图 11 各算法搜索总时间对比

## 4 结束语

针对当前分区过滤方法中过滤耗时长、分区大小不均衡和候选集规模大等问题,该文对图相似搜索算法进行改进,提出了多级过滤和分区初始顶点选择的

改进策略。通过多级过滤的方式,依次采用图大小、标签、VELG 以及分区索引进行逐级过滤,在大幅缩短搜索算法中过滤阶段时间的同时,降低了分区索引的大小和构建代价。采用分区初始顶点选择算法,保证最终分区尽量均衡,提升了分区过滤的下界紧密性,从而有效降低了候选图规模。在真实数据集上的对比实验表明,该算法在更低的索引代价下不仅缩短了过滤阶段时间、减小了候选集规模,而且提升了图相似搜索的总体效率。

**参考文献:**


[1] WHANG S E, ROH Y, SONG H, et al. Data collection and quality challenges in deep learning: a data-centric ai perspective[J]. The VLDB Journal, 2023, 32(4): 791-813.

[2] ROY-HUBARA N, STURM A, SHOVAL P. Designing NoSQL databases based on multiple requirement views[J]. Data & Knowledge Engineering, 2023, 145: 102149.

[3] DEUTSCH A, FRANCIS N, GREEN A, et al. Graph pattern matching in GQL and SQL/PGQ[C]//Proceedings of the 2022 international conference on management of data. Philadelphia: IEEE, 2022: 2246-2258.

[4] GUPTA P, MHEDHBI A, SALIHOGLU S. Columnar storage and list-based processing for graph database management systems[J]. arXiv: 2103.02284, 2021.

[5] 程　哲. 基于编辑距离的图相似性查询方法研究[D]. 西安:西安电子科技大学, 2022.

[6] ZHANG F, ZHANG Y, QIN L, et al. Efficiently reinforcing social networks over user engagement and tie strength[C]// 2018 IEEE 34th international conference on data engineering (ICDE). Paris: IEEE, 2018: 557-568.

[7] STAUFFER M, MAERGNER P, FISCHER A, et al. Cross-evaluation of graph-based keyword spotting in handwritten historical documents[C]//International workshop on graph-based representations in pattern recognition. Cham: Springer, 2019: 45-55.

[8] GENG C, JUNG Y, RENAUD N, et al. iScore: a novel graph kernel-based function for scoring protein - protein docking models[J]. Bioinformatics, 2020, 36(1): 112-121.

[9] 王忠庆. 基于编辑距离的图搜索问题研究[D]. 哈尔滨:黑龙江大学, 2021.

[10] 张军伟. 面向图相似性搜索的图编辑距离算法研究[D]. 桂林:桂林电子科技大学, 2023.

[11] GOUDA K, HASSAAN M. CSI_GED: an efficient approach for graph edit similarity computation[C]//2016 IEEE 32nd international conference on data engineering (ICDE). Helsinki: IEEE, 2016: 265-276.

[12] CHEN X, HUO H, HUAN J, et al. An efficient algorithm for graph edit distance computation[J]. Knowledge-Based Systems, 2019, 163: 762-775.

[13] CHANG L, FENG X, LIN X, et al. Speeding up GED verification for graph similarity search[C]//2020 IEEE 36th international conference on data engineering (ICDE). Dallas: IEEE, 2020: 793- 804.

[14] ZHAO X, XIAO C, LIN X, et al. Efficient structure similarity searches: a partition-based approach[J]. The VLDB Journal, 2018, 27(1): 53-78.

[15] LIANG Y, ZHAO P. Similarity search in graph databases: a multi-layered indexing approach[C]//2017 IEEE 33rd international conference on data engineering (ICDE). San Diego: IEEE, 2017: 783-794.

[16] 邱　珍, 郑朝晖. 高效低索引的图相似性搜索算法[J]. 计算机科学, 2023, 50(9): 130-138.