DOI: 10.16725/j.1673-808X.202349
引用本文: 龙广兴, 徐周波, 王嘉鑫. 一种新的多子图匹配符号算法[J]. 桂林电子科技大学学报, 2026, 46(3): 295-302.
Long Guangxing, Xu Zhoubo, Wang Jiaxin. A new symbolic algorithm for multiple subgraph matching[J]. Journal of Guilin University of Electronic Technology, 2026, 46(3): 295-302.

一种新的多子图匹配符号算法

龙广兴，徐周波，王嘉鑫

（桂林电子科技大学 计算机与信息安全学院, 广西 桂林 541004）

摘 要： 针对多子图匹配查询中两两相似性检测的时间复杂度高以及存储查询中间结果的空间复杂度高的问题, 提出了一种新的多子图匹配符号算法。通过引入图神经网络技术对查询图集合进行预分组, 降低图相似性检测复杂度; 利用代数决策图对查询图和数据图进行存储表示降低存储空间复杂度。为进一步提高有效共享子图检测以及查询顺序的生成的效率, 提出了一种新型最大公共连通子图算法。最后, 利用改进的子图匹配符号算法求解多子图匹配问题。实验结果表明, 与多子图同构查询优化算法相比, 所提的多子图匹配符号算法有效提高了多子图匹配的求解效率。

关键词： 多查询优化; 子图同构; 代数决策图; 图神经网络; 图匹配
中图分类号： TP391.4 **文献标志码：** A **文章编号：** 1673-808X(2026)03-0295-08

A new symbolic algorithm for multiple subgraph matching

LONG Guangxing，XU Zhoubo，WANG Jiaxin

（School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China）

Abstract: To address the high computational cost of pairwise graph similarity computation and the large memory overhead caused by storing intermediate query results in multi-subgraph matching, an efficient symbolic algorithm is proposed. Graph neural network(GNN) techniques are employed to pre-cluster query graphs and data graphs, thereby reducing the computational complexity of pairwise graph similarity computation. Furthermore, algebraic decision diagrams(ADDs) are adopted to compactly represent query graph sets and data graphs, thereby reducing memory consumption. To further improve shared subgraph detection efficiency and optimize query ordering, a novel maximum common connected subgraph(MCCSG) algorithm is developed. Finally, an improved subgraph matching algorithm, termed SSMGNN-MQO, is designed to solve the multi-subgraph matching problem. Experimental results demonstrate that the proposed method achieves significantly higher matching efficiency than the existing MQO-based subgraph isomorphism search method.

Key words: multi-query optimization; subgraph isomorphism; algebraic decision diagram; graph neural network; graph matching

图作为一种数据结构, 可有效刻画事物之间的关系。现实生活中许多复杂问题, 如社交网络、计算生物学和化学等领域都可以用图结构进行抽象表示[1]。而子图同构作为图数据分析中的基础问题, 具有重要研究意义。通常子图同构是指在数据图中查询所有与查询图同构的子图, 但由于子图同构问题属于 NP 问题[2], 随着数据规模的增长, 算法复杂度也呈指数增长。目前大量学者主要采用改进的剪枝规则和匹配顺序或引入基于图压缩[3] 的方法对相同属性的节点进行合并压缩来提高子图同构算法效率, 如 TurboIso[4]、CFL-Match[5]、VF3[6]、CECI[7]、SSMGNN[8]、SLF[9] 等算法。其中 SSMGNN 算法利用图神经网络聚合节点的邻域信息过滤查询图的节点候选集, 并利用 ADD 符号操作在数据图中构建多个候选分组, 有效

收稿日期: 2023-10-30
基金项目: 国家自然科学基金 (61762027)
通信作者: 徐周波 (1976—), 女, 教授, 博士, 研究方向为图数据管理、约束编程、符号计算。E-mail: xzbli_11@guet.edu.cn

减少冗余搜索，提升了搜索效率。然而，这些子图同构算法都只关注单个查询的效率提升，而未考虑多个查询同时处理。

在现实生活中的大部分应用场景，用户可能不知道查询图的确切结构，为提高查询的命中率，提交一组可能的查询图集合，如在犯罪侦查中，一组犯罪嫌疑人可以在社交网络中形成多种交互模式，警方在不知道嫌疑人确切联络关系的情况下提交一组可能的联络关系图集进行查询[10]；在化学分子式检索[11]中，查询图集合通常需要在数据库中进行批量子图匹配以提高查询效率。传统的单子图同构查询需对查询对象逐个处理，已无法满足这些应用场景，因此，针对多子图匹配问题展开研究。多子图匹配是指给定一个数据图 G 和一组查询图 $Q=\{q_1, q_2,\cdots, q_n\}$，对 $\forall q_i \in Q$，在数据图 G 中找到 q_i 的所有同构子图。由于现实生活中的真实查询集往往集中于图的某些热点区域，查询图之间存在相似关系，利用查询图之间的重叠子图可以加速查询，有效提升算法效率，但也存在以下问题：1) 如何快速检测出查询图之间的相似性，并对查询图进行相似性分组，降低多子图匹配的规模；2) 当查询图之间有明显重叠 (即共享子图) 时，如何对共享子图进行复用，加速算法的执行效率，降低缓存规模。

针对以上问题，Ren 等[12]提出了多子图同构查询优化(MQO)算法。MQO 算法首先通过对 n 个输入查询图作线下两两相似度计算，将计算结果存入相似性矩阵，相似度计算式为

$$\mathrm{GF}(q_i,q_j)=\frac{\min(\mathrm{LI}(q_i,f_{\mathrm{TLS}}(q_i,q_j)),\mathrm{LI}(q_j,f_{\mathrm{TLS}}(q_i,q_j)))}{\min(S_{\mathrm{TLS}}(q_i),S_{\mathrm{TLS}}(q_i))},\tag{1}$$

其中：TLS(Tri-Vertex Label Sequence) 为图的三节点标签序列；$f_{\mathrm{TLS}}(q_i, q_j)$ 表示 q_i 和 q_j 共享的 TLS 集合；$\mathrm{LI}(q_i, f_{\mathrm{TLS}}(q_i, q_j))$ 表示在共享的 TLS 集合中构造 q_i 的最大公共连通子图所需要的 TLS 实例个数；$S_{\mathrm{TLS}}(q_i)$ 表示 q_i 中所有 TLS 的实例个数之和。

然后算法对相似性矩阵按人为设定的阈值进行团划分，大于阈值的查询图对划入一个团中。对相似性团中的查询图两两计算最大公共子图来构建 PCM(pattern containment map) 结构，并按自顶向下的顺序实现同构查询，保证父节点缓存结果被有效共享。当查询图与其他所有查询图的相似性值小于给定阈值时，则按单子图同构算法逐个处理。由于 MQO 算法中相似度计算需要对每个 TLS 统计实例个数，计算出两图相交的 TLS，并统计由相交的 TLS 的实例集合组成的最大连通图子图所使用的实例个数，致使算法时间复杂度高。同时每个 TLS 实例都需要缓存以用于查询图之间的反复计算，致使空间复杂度高。此外，传统数据结构在保存已查询子图的同构解集，存储压缩效率低，即使 MQO 算法中使用了图划分技术也不可避免内存使用过大问题。孙云浩等[13]提出了 M-PM 算法，对 MQO 算法的 PCM 结构进行扩展，提出了节点分片表的概念，在 RDF 数据集上的运行时间比 MQO 算法提升了 30%，但该算法在其他数据集上不能一直保持优势。

为提高多子图匹配的求解效率和降低缓存空间规模，引入了图神经网络[14-16]和代数决策图 (ADD)[17]。由于图神经网络具有强大的学习表示能力，通过迭代训练图神经网络模型能有效学习表示图的结构和属性信息的特征向量，进而实现快速图分类。而符号 ADD 技术则具有高效紧凑且易于操作等特性[17]，利用 ADD 对图进行隐式表示，通过符号操作实现并行约束搜索，在压缩存储空间的同时可有效提升算法执行效率。因此，提出了一种新的多子图匹配符号 (DDMQO) 算法。在多子图匹配中，引入图表示学习和图神经网络技术，根据查询图的相似性进行预先分组，减少最大公共子图计算；使用高效紧凑易操作的 ADD 对图结构进行表示，减少问题的空间需求，减缓状态爆炸；提出了最大公共连通子图符号 (DDMCS) 算法，并改进子图匹配符号 (SSMGNN) 算法[8]，在缩减多查询搜索空间的同时加快问题的求解效率。

1 相关概念

定义 1 带标签无向图 $G=\langle V, E, L, l\rangle$，其中：$V$ 表示图 G 的节点集合；E 表示图 G 的边集，且 $\forall e \in E$ 为无向边；L 表示节点的标签集合；l 表示节点与标签之间的映射函数，$V\rightarrow L$。

定义 2 给定查询图 $q=\langle V_q, E_q, L_q, l_q\rangle$ 和数据图 $G=\langle V_G, E_G, L_G, l_G\rangle$，图 q 到图 G 存在子图同构关系，当且仅当存在一个单射函数 f: $V_q\rightarrow V_G$ 满足：

1) $\forall u\in V_q$, $\exists f(u)\in V_G$, $l_q(u)=l_G(f(u))$；

2) $\forall (u,u')\in E_q$, $\exists (f(u), f(u'))\in E_G$。

定义 3 已知图 G 和 G'，若存在另外一个图 g，满足 $g\subseteq G$, $g\subseteq G'$，且不存在图 g' ($g'\subseteq G$, $g'\subseteq G'$)，使得 $|g'|\geqslant|g|$，则称 g 是图 G 和 G'的最大公共导出子图，记为 MCIS(G, G')。

图 1 所示的图 G 和 G'，它们的最大公共子图为 g=MCIS(G, G')。

定义 4 ADD[17]是表示基于变量序 π: $x_1<x_2<\cdots<x_n$ 的一簇伪布尔函数 f_i: $\{0,1\}^n\rightarrow S$ 的一个有向无环图，其满足：

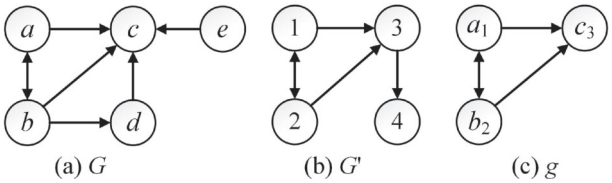


图 1　图 G、G'和 g

1)S 为 ADD 代数结构的有限值域, 且 $S\in\mathbf{Z}$。

2)ADD 中的节点分为根节点、终节点和内部节点三类。

3)终节点集合记为 T。对任意 $t\in T$, 均被标识为值域 S 中的一个元素 $s(t)$。每个非终节点 u 具有四元组属性 (f^u, n_{var}, l, h), 其中: f^u 表示节点 u 所对应的伪布尔函数; n_{var} 表示节点 u 的标记变量; l 表示节点 u 的 $n_{var}=0$ 时, 节点 u 的 0-分支子节点; h 表示节点 u 的 $n_{var}=1$ 时, 节点 u 的 1-分支子节点。

4)图中的每个节点 u 对应唯一一个函数 f^u。

5)图中任意一条从根节点到终节点的路径中, 所有变量均按变量序 π 的顺序出现且仅出现一次。

2　带标签无向图的 ADD 表示

给定带标签无向图 G, 图的节点和标签数量分别为 $|V|=a$ 和 $|L|=b$。传统的数据结构如邻接矩阵需要 $|V|\times|V|$ 的存储空间对图进行储存。为了对图 G 进行 ADD 压缩存储, 首先要对图 G 中的节点集、标签集和边集进行伪布尔函数表示:

1)节点集 V: 对 V 中的各节点用 $X=\{x_0, x_1, \cdots, x_{k-1}\}\in\{0,1\}^k$ 进行编码表示, 其中 $k=\lceil\log_2 a\rceil$, 每个节点 $v_i\in V$ 对应唯一一个由 k 个变量表示的极小项 $m_i(x_0, x_1, \cdots, x_{k-1})$, 则节点集 V 可由伪布尔函数 $f_V(x_0, x_1, \cdots, x_{k-1})=m_1(x_0, x_1, \cdots, x_{k-1})+m_2(x_0, x_1, \cdots, x_{k-1})+\cdots+m_a(x_0, x_1, \cdots, x_{k-1})$ 表示。

2)标签集 L: 对 L 中的各个标签用 $U=\{u_0, u_1, \cdots, u_{r-1}\}\in\{0,1\}^r$ 进行编码表示, 其中 $r=\lceil\log_2 b\rceil$, 每个标签 $l_j\in L$ 对应唯一一个由 r 个变量表示的极小项 $m_j(u_0, u_1, \cdots, u_{r-1})$, 则标签集 L 可由伪布尔函数 $f_L(u_0, u_1, \cdots, u_{r-1})=m_1(u_0, u_1, \cdots, u_{r-1})+m_2(u_0, u_1, \cdots, u_{r-1})+\cdots+m_b(u_0, u_1, \cdots, u_{r-1})$ 表示。

3)边集 E: 对于 $\forall e\in E$, 由 $(U, X)=\{u_0, u_1, \cdots, u_{r-1}, x_0, x_1, \cdots, x_{k-1}\}$ 和 $(V, Y)=\{v_0, v_1, \cdots, v_{r-1}, y_0, y_1, \cdots, y_{k-1}\}$ 分别对 e 的 2 个端点进行编码表示, 则边集 E 可表示为伪布尔函数:

$$f_E(U,X,V,Y)=\begin{cases}1, e\in E;\\ 0, e\notin E。\end{cases}$$

例如, 对于图 2 中的图 2(a), 节点集 $\{v_1, v_2, v_3\}$ 和标签集 $\{A, B\}$ 分别由 $X=\{x_0, x_1\}$、$U=\{u_0\}$ 编码表示, 其中节点 v_1 的小项 $x_0'x_1'$, 各节点及标签对应的小项如图 2(b) 所示。边集 E 的伪布尔函数 f_E 对应的 ADD 表示如图 2(c) 所示。

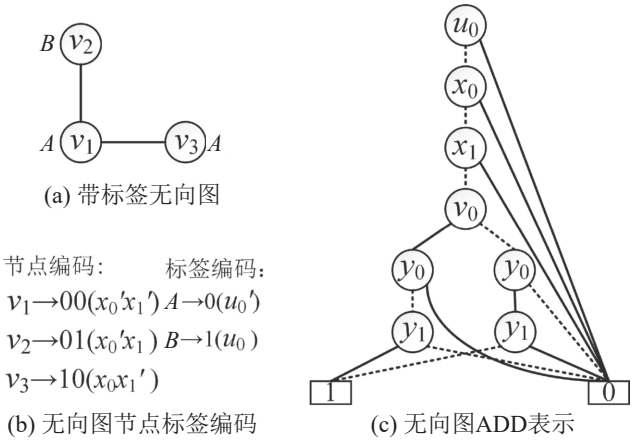


图 2　带标签无向图的 ADD 表示

3　多子图匹配符号算法 DDMQO

DDMQO 算法的基本框架和伪代码分别如图 3 和算法 1 所示。

算法 1　DDMQO。

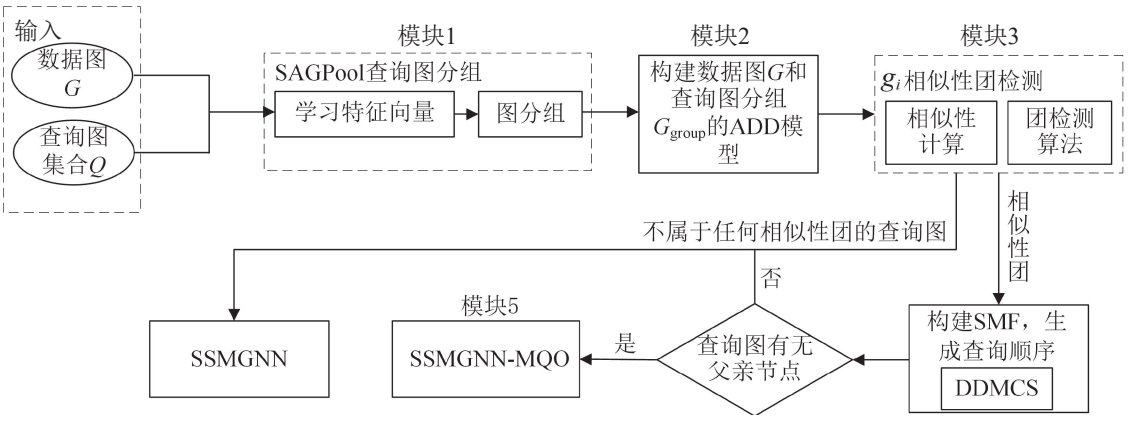


图 3　DDMQO 算法的基本框架

输入: 查询图集合 $Q=\{q_1, q_2, \cdots, q_n\}$, 数据图 G。

输出: q_i 在 G 中的所有同构子图, $\forall q_i \in Q$。

1. G_{group} = SAGPool(Q);
2. $Q_{\text{vector}}[n](U, X, V, Y)$, $E(U, X, V, Y)$ = ADDBuild (G_{group}, G);
3. for $Q_{\text{vector}}[i](U, X, V, Y) \in Q_{\text{vector}}[n](U, X, V, Y)$ do
4. (G_{vector}, G_{ns}) = SimGroupDetection($Q_{\text{vector}}[i](U, X, V, Y)$);
5. SMF = SMFBuilid(G_{vector});
6. G_{order} = QueryExecutionOrder(SMF);
7. for each $q_i \in G_{\text{order}}$ do
8. if q_i has parents then
9. q_i= SSMGNN-MQO(q_i, $G(U, X, V, Y)$);
10. for each parent q_j of q_i do
11. if $q_j \notin Q$ and all child of q_j are arrived
12. clear(q_j);
13. else
14. q_i = SSMGNN(q_i, $G(U, X, V, Y)$);
15. for each $q_i \in G_{\text{ns}}$ do
16. q_i= SSMGNN(q_i, $G(U, X, V, Y)$)。

算法 1 中第 1 行对应图 3 的模块 1, 利用 SAGPool 图神经网络模型[14] 对查询图进行相似性分类。第 2 行对应模块 2, 将分类后的分组集合 G_{group} 和数据图 G 按第 4 节所示进行 ADD 表示, 其中 n 表示查询图集合的分组数, n 的大小取决于数据图的特性。第 4 行对应模块 3, 对 G_{group} 中每个 g_i 进行组内相似性检测和团划分, 将高于相似性阈值的查询图划入一个团中, G_{vector} 为存放相似性团的容器, 与任何团均不存在相似性的查询图存放在 G_{ns}, 并直接调用 SSMGNN 算法[8] 进行子图查询。第 5、6 行对应模块 4, 对 G_{vector} 中的团进行最大公共连通子图计算, 检测出共享子图, 将其与原查询图组织起来并添加到 SMF(subgraphs map forests)中, 将共享子图标记为对应查询图的父节点, 自底向上形成森林结构。计算查询的执行顺序, 确保共享子图的查询结果可以复用, 并尽可能早地释放无用缓存。最后第 7~16 行, 对 SMF 中的每个查询图, 调用改进的 SSMGNN-MQO 算法求解, 该算法可以利用其父节点的缓存结果加速搜索。对于 q_i 的每个父节点 q_j, 若 q_j 的所有子节点都被已处理, 则释放 q_j 的缓存。对于无父节点的 q_i 和 G_{ns} 中的查询图, 直接调用 SSMGNN 算法进行计算。

3.1 相似性团检测

相似性团是指查询图中相似性较高的图集合。一组图集中存在相似关系的图集合可能有多个, 构成多个相似性团, 一个图也可以同时存在于多个相似性团中。为提高查询图之间相似性计算的效率, 图相似性计算式为

$$S(Q,P)=\frac{\min(N(k(U,V,W)\wedge q(U,X,V,Y,W,Z)),N(k(U,V,W)\wedge p(U,X,V,Y,W,Z)))}{\min(q(U,X,V,Y,W,Z),p(U,X,V,Y,W,Z))}。$$

其中, $q(U, X, V, Y, W, Z)$ 和 $p(U, X, V, Y, W, Z)$ 是查询图 Q 和 P 的三节点标签序列的 ADD 表示。

基于第 4 节可知边集 E 的 ADD 表示, 其实质是二节点标签序列集合。增加 W, Z 变量用于表示与 (V, Y) 相连接的下一个节点, 可得图的三节点标签序列的 ADD 表示。查询图 Q 和 P 重叠的三标签序列 $k(U, V, W) = (\exists X)(\exists Y)(\exists Z)q(U, X, V, Y, W, Z) \wedge (\exists X)(\exists Y)(\exists Z)p(U, X, V, Y, W, Z)$。$N(U, X, V, Y, W, Z)$ 用于表示 ADD 结构中的小项个数, 即路径总数。

与式 (1) 相比, 在符号操作下该计算公式效率更高。相似团检测的伪代码如算法 2 所示。

算法 2 相似性团检测。

输入: 查询图组 $Q_{\text{vector}}[i](U, X, V, Y)$。

输出: 相似性团组 G_{vector}, G_{ns}。

1. for each $q_i(U, X, V, Y)$, $q_j(U, X, V, Y) \in Q_{\text{vector}}[i](U, X, V, Y)$
2. $S_{\text{TVLSqi}}(U, X, V, Y, W, Z)$ = getTVLS($q_i(U, X, V, Y)$);
3. $S_{\text{TVLSqj}}(U, X, V, Y, W, Z)$=getTVLS($q_j(U, X, V, Y)$);
4. $S_{\text{TLSqi}}(U, V, W)$ = getTLS($S_{\text{TVLSqi}}(U, X, V, Y, W, Z)$);
5. $S_{\text{TLSqj}}(U, V, W)$ = getTLS($S_{\text{TVLSqj}}(U, X, V, Y, W, Z)$);
6. $p(U, V, W)$ = OverPath($k(U, V, W)$, $S_{\text{TLSqi}}(U, V, W)$);
7. $q(U, V, W)$ = OverPath($k(U, V, W)$, $S_{\text{TLSqj}}(U, V, W)$);
8. (N_p, N_q, N_{qi}, N_{qj}) = CountPath(p, q, S_{TLSqi}, S_{TLSqj});
9. $a_{\text{threshold}}[i][j] = \min(N_p, N_q)/\min(N_{qi}, N_{qj})$;
10. (G_{vector}, G_{ns}) = ClusterCheck($a_{\text{threshold}}[i][j]$, σ)。

算法 2 首先对输入的查询图组 $Q_{\text{vector}}[i](U, X, V, Y)$ 进行遍历, 通过 getTVLS 函数得到 q_i 和 q_j 的三节点标签序列 S_{TVLSqi} 和 S_{TVLSqj}, 符号操作:

$$S_{\text{TVLSq}}(U, X, V, Y, W, Z) = q(U, X, V, Y) \wedge q(V, Y, W, Z), \quad (2)$$

再通过第 4、5 行 getTLS 函数得到 q_i 和 q_j 的三标签序列 S_{TLSqi} 和 S_{TLSqj}, getTLS 函数实质是对传入参数存在量化掉 X、Y、Z 节点变量, 然后通过第 6~7 行 OverPath 函数计算 2 个三标签序列的重叠路径, 具

体操作如式 (3)~(5) 所示。

$$K(U, V, W) = S_{\text{TLSqi}}(U, V, W) \wedge S_{\text{TLSqj}}(U, V, W), \quad (3)$$

$$K(U, V, W) = 01\text{ADD}(K(U, V, W)), \quad (4)$$

$$P_{\text{overPath}}(U, V, W) = K(U, V, W) \wedge S_{\text{TLSqi}}(U, V, W)。 \quad (5)$$

其中, 函数 01ADD($K(U, V, W)$) 表示将 $K(U, V, W)$ 转换成仅含有 0 和 1 终节点的 ADD 图, 即大于 1 的终节点值均改为 1。

第 8 行中函数 CountPath 统计 N_p、N_q、N_{qi} 和 N_{qj} 等各 ADDs 图中三标签序列的具体实例数。第 9 行对图 P 和 Q 计算相似性, 并存入相似性矩阵 $\boldsymbol{a}_{\text{threshold}}$ 中。最后通过团检测函数将相似度高于阈值 σ 的查询图划入同一个团中, G_{vector} 表示多个团的集合。同时将与组内任意图都不存在相似关系的查询图存入 G_{ns} 中。

3.2　构建子图映射森林（SMF）

SMF 结构定义: 给定一个查询图集合 $Q=\{q_1, q_2, \cdots, q_n\}$, 一个子图映射森林结构(subgraphs map forests)是一个有向图 $P_{\text{Q}}=\{V_{\text{SMF}}, E_{\text{SMF}}\}$, 其中每一个带标签无向图 $q_{\text{SMF}} \in V_{\text{SMF}}$ 满足以下关系。

1)$\forall q_i \in Q$, $\exists q_{\text{SMF}} \in V_{\text{SMF}}$, 使得 $q_i = q_{\text{SMF}}$;

2)$\forall q_{\text{SMF}} \in V_{\text{SMF}}$, $\exists q_i \in Q$, 使得 $q_{\text{SMF}} \subseteq q_i$;

3)在 V_{SMF} 中不存在 2 个节点 q 和 q', 使得 $q \cong q'$;

4)当且仅当 $q \subseteq q'$, 并且没有 $q'' \in V_{\text{SMF}}$, 使得 $q' \neq q''$, $q \subseteq q''$和 $q'' \subseteq q'$, 则存在一条有向边 $(q, q') \in E_{\text{SMF}}$。

由于相似性团之间查询图可能存在交集, 每个相似性团内进行两两计算最大公共连通子图会出现重复计算, 且计算量大。因 ADD 图是伪布尔函数的规范型表示, 且通过合并规则和删除规则, 使得查询图中相同的子图均共享同一个 ADD 存储子结构, 为此, 在 MQO 算法的基础上增加了 V_{edgePair} 集合用于存储已经计算过的查询图对, 最大限度避免重复计算。SMF 构建的算法伪代码如算法 3 所示。

算法 3　构建子图映射森林。

输入: 查询图集合 Q。

输出: 子图映射森林 SMF。

1. $V_{\text{edgePair}} \leftarrow \varnothing$;
2. for each $g_i \in G_{\text{vector}}$ do
3. $\quad G_{\text{nextLevel}} \leftarrow \varnothing$;
4. $\quad$ create a V_{SMF} for each query in g_i;
5. $\quad$ divide queries in g_i into disjoint pairs (q_i, q_j);
6. $\quad$ if $(q_i, q_j) \in V_{\text{edgePair}}$
7. $\quad\quad$ goto line 5;
8. $\quad$ else $V_{\text{edgePair}} \leftarrow (q_i, q_j)$;
9. $\quad$ for each random pairs $(q_i, q_j) \in g_i$ do
10. $\quad\quad Q_{\text{MCS}} \leftarrow$ DDMCS($q_i(U, X, V, Y)$, $q_j(U, X, V, Y)$);
11. $\quad\quad$ for each $q_{\text{MCS}} \in Q_{\text{MCS}}$ do
12. $\quad\quad$ if $q_{\text{MCS}} \cong q_i$(assuming $|q_i| \leqslant |q_j|$) then
13. $\quad\quad\quad$ add q_i into $G_{\text{nextLevel}}$;
14. $\quad\quad\quad$ add q_j into children of q_i;
15. $\quad\quad$ else
16. $\quad\quad\quad$ add q_{MCS} into $G_{\text{nextLevel}}$;
17. $\quad\quad\quad$ add both q_i, q_j into children of q_{MCS};
18. $\quad$ if $|G_{\text{nextLevel}}|>1$ then
19. $\quad\quad g_i \leftarrow G_{\text{nextLevel}}$;
20. $\quad\quad$ goto line 3;
21. MergeIsomorphicNodes(SMF);
22. TransitiveReducion(SMF);
23. return SMF。

在算法 3 中首先初始化 V_{edgePair} 集合为空集(行 1), 然后对每个相似性团中的每个查询创建 SMF 节点(行 4)。将每个相似性团中的图随机分成不重复的图对, 如果该图对在之前已经计算过, 则重新划分, 否则将新图对存入 V_{edgePair} 集合中(如果查询图数量为奇数, 将最后一个查询图直接放入 $G_{\text{nextLevel}}$ 中)(行 5~8)。第 9~17 行计算 V_{edgePair} 每个图对的 MCSs, 将 MCSs 放入 $G_{\text{nextLevel}}$, 并添加相应的 SMF 边。对 $G_{\text{nextLevel}}$ 中的查询重复第 3~20 行的计算过程, 直至只剩下一个查询(行 18~20)。最后, 对同构 SMF 节点进行合并, 并去除冗余边(行 21~23)。

由于使用 ADD 对查询图进行表示, 传统的最大公共子图求解算法已无法满足需求, 因此提出了基于符号技术的最大公共连通子图求解算法 DDMCS(行 10), 使用高效紧凑且易操作的 ADDs 对图结构进行隐式表示, 通过集合间的并行约束处理加快搜索速度, 减少问题的空间需求。DDMCS 算法伪代码如算法 4 所示。

算法 4　DDMCS。

输入: 查询图 $q(U, X, V, Y)$, 数据图 $p(U, X, V, Y)$。

输出: 最大公共连通子图 Q_{MCS}。

1. $q_{\text{vSet}}(U, X)$, $p_{\text{vSet}}(U, X)$ = GetNode($q(U, X, V, Y)$, $p(U, X, V, Y)$);
2. $q_{\text{s}}(U, X, V, Y)$, $p_{\text{s}}(U, X, V, Y)$ = SimpleGraph($q(U, X, V, Y)$, $p(U, X, V, Y)$, $q_{\text{vSet}}(U, X)$, $p_{\text{vSet}}(U, X)$);
3. for each $q_i(U, X) \in q_{\text{vSet}}(U, X)$ do
4. $\quad C(U, X, V, Y)$ = InitCand($q_i(U, X)$, $p_{\text{vSet}}(U, X)$);
5. Q_{MCS} = MCS($q(U, X, V, Y)$, $p(U, X, V, Y)$, $C(U,$

$X, V, Y)$, $V_{\text{mcsOrder}}[n](U, X)$, solution);

6. return Q_{MCS}。

Subroutine MCS($q(U, X, V, Y)$, $p(U, X, V, Y)$, $C(U, X, V, Y)$, $V_{\text{mcsOrder}}[n](U, X)$, solution)

1. static l++;
2. if $C(U, X, V, Y)$= =0 return solution;
3. $k(U, X, V, Y) = V_{\text{mcsOrder}}[l](U, X) \wedge C(U, X, V, Y)$;
4. $C_{\text{New}}(U, X, V, Y) = C(U, X, V, Y) - k(U, X, V, Y)$;
5. for each $q_i(U, X) \in M_{\text{visitedSet}}(U, X)$ do
6. if ($q_i(U, X) \wedge V_{\text{mcsOrder}}[l](V, Y) \wedge q(U, X, V, Y) \neq 0$) and C and($q_i$) $\wedge$ C and ($V_{\text{mcsOrder}}[l]$) $\wedge p(U, X, V, Y) = = 0$)
7. cost++;
8. if cost>solution.cost return;
9. $M_{\text{visitedSet}}(U, X) \leftarrow V_{\text{mcsOrder}}[l](U, X)$;
10. MCS($q(U, X, V, Y)$, $p(U, X, V, Y)$, $C_{\text{New}}(U, X, V, Y)$, $V_{\text{mcsOrder}}[n](U, X)$, solution)。

在算法 4 中，首先得到图 p、q 的节点集 $q_{\text{vSet}}(U, X)$, $p_{\text{vSet}}(U, X)$（第 1 行）。再通过 SimpleGraph 函数剔除不存在相同标签的节点，得到简化后的图 $q_{\text{s}}(U, X, V, Y)$、$p_{\text{s}}(U, X, V, Y)$（第 2 行）。SimpleGraph 函数可通过式 (6)~(8) 实现。

$$L_s(U) = (\exists X) q_{\text{vSet}}(U, X) \wedge (\exists X) p_{\text{vSet}}(U, X), \tag{6}$$

$$q_s(U, X, V, Y) = L_s(U) \wedge L_s(V) \wedge q(U, X, V, Y), \tag{7}$$

$$p_s(U, X, V, Y) = L_s(U) \wedge L_s(V) \wedge p(U, X, V, Y)。 \tag{8}$$

对查询图 q_{s} 中的每个节点 q_i, InitCand 函数是在数据图 p_{s} 中找出与 q_i 标签相同的节点划入 q_i 的初始候选集，并将 q_i 的候选集并入总的候选集 C 中（第 4 行）。InitCand 函数可通过式 (9)~(11) 实现。

$$q_i(V) = (\exists Y)(q_i(V, Y)), \tag{9}$$

$$C_{qi}(U, X, V, Y) = q_i(U, X) \wedge (q_i(V) \wedge p_{\text{vSet}}(V, Y)), \tag{10}$$

$$C(U, X, V, Y) = C(U, X, V, Y) \vee C_{qi}(U, X, V, Y)。 \tag{11}$$

最后调用最大公共子图函数进行递归求解，返回结果为两图的最大公共子图（第 5、6 行）。

在子函数 MCS 求解过程中为尽早过滤掉不满足的解，加快算法求解速度，在回溯过程中引入了 cost 代价值，为不匹配的节点更新目标值的上界和下界。在递归求解的过程中，cost 值会不断更新，不断的保留最小的 cost 值，一旦发现当前解的 cost 值大于已保留的最小 cost 值则停止搜索并放弃这组解，否则对解和最小 cost 值进行更新。

3.3 子图同构嵌入优化算法 SSMGNN-MQO

基于 SSMGNN 算法[8] 和 SMF，提出一种改进的子图同构搜索方法 SSMGNN-MQO。在 SMF 中，对于只有一个父节点的查询图，直接将父节点中的解集映射到子节点对应的查询节点上，对不存在映射的子节点继续运行 SSMGNN 算法求解；若存在多个父节点，则取所有父节点的最大交集的节点集合映射到子节点的对应节点上。对于父节点中各自不相交的查询节点，分别映射到子节点上，用父节点的缓存嵌入替换子节点中查询节点的候选集，缩减候选集的规模，再继续运行 SSMGNN-MQO 算法求解。子图同构嵌入优化算法伪代码如算法 5 所示。

算法 5 SSMGNN-MQO。

输入：查询图 $q(U, X, V, Y)$，数据图 $G(U, X, V, Y)$，查询图 q 在 SMF 中的所有父亲节点集合 Parent(q)。

输出：子图匹配的所有解 S。

1. $C(U, X, V, Y)$ = EmbedGraph(q, Parent(q));
2. V_{order}, $W_{\text{root}}(U, X)$ = Order($q(U, X, V, Y)$, $C(U, X, V, Y)$);
3. $R(Z, U, X, V, Y)$=ADDPartition($G(U, X, V, Y)$, $q(U, X, V, Y)$, $C(U, X, V, Y)$, $W_{\text{root}}(U, X)$);
4. $C_{\text{root}}(U, X)$ = RootCand($C(U, X, V, Y)$, $W_{\text{root}}(U, X)$);
5. for $V_{\text{root}}(U, X) \in C_{\text{root}}(U, X)$
6. $R_i(U, X, V, Y)$=ChooseRegion($C(U, X, V, Y)$, $R(Z, U, X, V, Y)$);
7. Match(V_{order}, $R_i(U, X, V, Y)$, $V_{\text{root}}(U, X)$, S);
8. return S。

Subroutine EmbedGraph(q, Parent(q));

1. $C(U, X, V, Y) \leftarrow \varnothing$;
2. $T_{\text{comEmbed}}(U, X, V, Y)$=ChooseEmbed($\cap_{f(U,X,V,Y) \in \text{Parent}(q)}$);
3. $C(U, X, V, Y) = C(U, X, V, Y) \vee T_{\text{comEmbed}}(U, X, V, Y)$;
4. for each $f(U, X, V, Y) \in$Parent(q) do
5. $C(U, X, V, Y)$=Update(ChooseEmbed($f(U, X, V, Y) - T_{\text{comEmbed}}(U, X, V, Y)$), $C(U, X, V, Y)$);
6. return $C(U, X, V, Y)$。

SSMGNN-MQO 算法的改进之处在于使用 EmbedGraph 函数对 SMF 结构中的父节点实现了解集映射，因此主要对 EmbedGraph 函数进行介绍。首先初始化 C 为空集，用于存放查询图 q 的嵌入候选集（子函数第 1 行）。然后取 q 的所有父节点的最大交集作为公共嵌入，直接映射到查询图 q 的对应查询节点上

(子函数第 2、3 行)。对于每个父节点的解集, 首先剔除已经嵌入过的节点, 然后利用 Update 函数将剩下的节点更新到 C 中, Update 函数需要对每个节点进行标记, 对未更新过的节点进行直接候选集映射, 对更新过的节点则将当前的候选集映射与 C 中已经存在的候选集取交集并更新(子函数第 4、5 行)。用更新后的 C 候选集替换 SSMGNN 算法中的候选集得到 SSMGNN-MQO 算法(第 1 行), 有效缩减了候选集搜索规模, 降低解的组合数量, 提高求解效率。

4　实验结果及分析

将 DDMQO 算法与 MQO 算法[11] 进行实验对比来评估 DDMQO 算法求解效率。实验方案: 1)生成一组查询图数据集, 通过图神经网络分类模型 SAGPool[14] 对查询图数据集进行训练, 提高查询图相似性分组效率; 2)将 DDMQO 算法与 MQO 算法进行实验对比, 验证 DDMQO 算法要优于 MQO 算法。

为对 SAGPool 进行训练, 使其能达到对图数据进行相似性分组的效果, 需针对特定的数据图类型制作相应的训练集。方法是在数据图上随机挑选 n 个不相关且节点度大的点作为核心节点。因为对于大多数有实际意义的图, 节点度越大, 表示节点越重要, 它的邻居结构受到用户的关注度也越高。对于每个核心节点, 通过随机游走得到包含 k 个节点的核心子图, 构成查询图数据集的基准图。最后在基准图上进行随机扩展, 得到查询图的数据集。通过此规则分别对 Human 和 Yeast 数据集生成了 1 000 个查询图, 查询图的大小为 8~15 个节点。

使用了 2 个基准数据集 Yeast 和 Human 来测试算法。这 2 个数据集具有不同的特征。Yeast 是一个数据节点较少但标签较多的图, Human 是一个节点度很大的社交网络图, 如表 1 所示。

表 1　Yeast 和 Human 数据集

数据集	节点数	边数	标签数	节点平均度
Yeast	3 112	12 915	184	8.05
Human	4 675	86 282	44	36.82

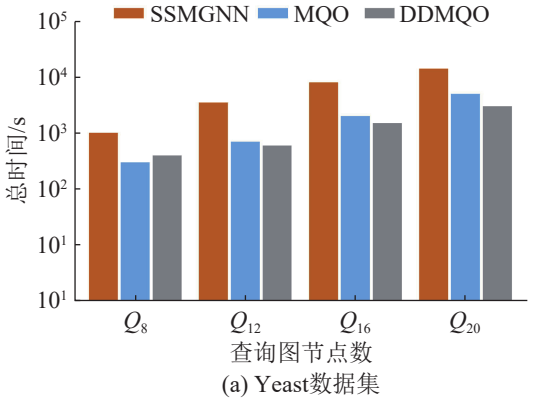
为有效检验 DDMQO 算法的性能, 利用 Colorado 大学的 CUDD 软件包, 在 Windows 10, Core-i5-6500@2.3 GHz CPU, 16 GiB RAM 上, 编译语言为 Python 与 C/C++。

使用 SAGPool 模型[14] 对查询图进行训练分组, 实验结果如表 2 所示, 对于不同属性的数据集, 分组精度不同。对于 Human 数据集, 由于其标签少且稠密, 通过图神经网络模型的多次迭代可以较好地学习到图的结构和属性信息。而对于 Yeast 数据集来说, 由于其稀疏且标签数多, 即便降低分组数量也很难得到较高的分组精度。在实验过程中发现, 在对图进行相似性分组时, 并非分组越多越好, 针对特定的图结构选择最佳分组数, 有助于进一步提高算法效率。

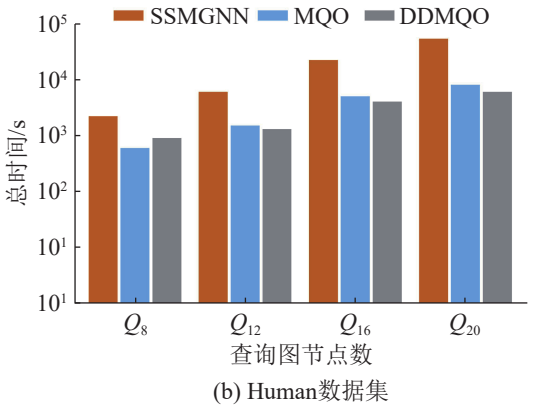
表 2　Yeast 和 Human 查询图数据集分组精度

数据集	分组数	节点数	边数	标签数	分组精度
Yeast	25	12	11.4	184	0.64
Human	40	12	20.6	44	0.72

多子图匹配的实验结果如图 4 所示。图 4(a) 表示 SSMGNN 算法、MQO 算法与 DDMQO 算法在 Yeast 数据集上的总运行时间; 图 4(b) 表示 3 种算法在 Human 数据集上的总运行时间。通过实验对比可知:



(a) Yeast数据集



(b) Human数据集

图 4　Yeast 与 Human 数据集的实验结果

1)当查询集为 Q_8 时, MQO 算法在 2 个数据集上的总求解时间较少, 优于本算法。其原因是在算法执行过程中, 构建 ADD 需要消耗一定时间, 且 DDMQO 算法利用图神经网络对图的相似性进行训练分组, 在查询图较小时, 其分组效率并不明显。然而, 随着查询图规模的增加, DDMQO 算法的总时间性能明显优于 MQO 算法, 且时间增幅小于 MQO 算法, 这表明本

算法在处理较大规模查询图时更具优势。

2)MQO 算法在 Human 数据集的求解时间比在 Yeast 数据集上耗时更长。其原因是,Human 数据集是稠密图且标签更少,MQO 算法在 Human 数据集需要花费更多的时间进行搜索求解。同样,DDMQO 算法在稠密上的求解时间也高于稀疏数据集。但是,DDMQO 算法求解时间并不与图的稠密程度呈显著相关性,它与对图进行 ADD 编码时的编码长度关联度更大。一般来说,ADD 图采用二进制编码,只要 2 个数据图节点的编码位数加上标签的编码位数长度相等,那么算法的求解时间就相近。

5 结束语

提出了一种结合图神经网络与符号技术解决多子图匹配问题的符号算法 DDMQO。该算法利用图神经网络对查询图进行相似性分组,提高了分组效率。算法引入了代数决策图(ADD)对图模型进行刻画,使其能够在大规模数据图上进行并行约束处理。还提出了基于符号技术的最大公共连通子图算法 DDMCS,并单子图同构算法 SSMGNN 进行了改进,使其可以复用父节点的同构嵌入。下一步,将研究如何使用符号技术和代数决策图求解流子图同构查询问题[18],为输入查询流提供更好的实时求解方案,提高求解效率。

参考文献:

[1] Han M, Kim H, Gu G, et al. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together[C]//Proceedings of the 2019 International Conference on Management of Data. New York, NY: ACM Press, 2019: 1429-1446.

[2] Conte D, Foggia P, Sansone C, et al. Thirty years of graph matching in pattern recognition[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2004, 18(3): 265-298.

[3] Ren Xuguang, Wang Junhu. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs[J]. Proceedings of the VLDB Endowment, 2015, 8(5): 617-628.

[4] Han W S, Lee J, Lee J H. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases[C]//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, NY: ACM Press, 2013: 337-348.

[5] Bi Fei, Chang Lijun, Lin Xuemin, et al. Efficient subgraph matching by postponing cartesian products[C]//Proceedings of the 2016 International Conference on Management of Data. New York, NY: ACM Press, 2016: 1199-1214.

[6] Carletti V, Foggia P, Saggese A, et al. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(4): 804-818.

[7] Bhattarai B, Liu H, Huang H H. Ceci: compact embedding cluster index for scalable subgraph matching[C]//Proceedings of the 2019 International Conference on Management of Data. New York, NY: ACM Press, 2019: 1447-1462.

[8] 杨欣, 徐周波, 陈浦青, 等. 基于图神经网络的子图匹配符号算法[J]. 桂林电子科技大学学报, 2022, 42(5): 391-397.

[9] Liang Wenle, Dong Wenyong, Yuan Mengting. SLF: a passive parallelization of subgraph isomorphism[J]. Information Sciences, 2023, 623: 900-914.

[10] Nataraian M. Undertsanding the structure of a drug traffficing organization: a converational analysis[J]. Crime Preventon Studies, 2000, 11: 273-298.

[11] Yan Xifeng, Yu P S, Han Jiawei. Substructure similarity search in graph databases[C]//Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. New York, NY: ACM Press, 2005: 766-777.

[12] Ren Xuguang, Wang Junhu. Multi-query optimization for subgraph isomorphism search[J]. Proceedings of the VLDB Endowment, 2016, 10(3): 121-132.

[13] 孙云浩, 李逢雨, 李冠宇, 等. 面向 RDF 图的多模式匹配方法[J]. 计算机工程与应用, 2020, 56(13): 84-92.

[14] Lee J, Lee I, Kang J. Self-attention graph pooling[J]. Statistics, 2019, 3: 3734-3743.

[15] 宁懿昕, 谢辉, 姜火文. 图神经网络社区发现研究综述[J]. 计算机科学, 2021, 48(202): 11-16.

[16] 徐周波, 李珍, 刘华东, 等. 基于邻居信息聚合的子图同构匹配算法[J]. 计算机应用, 2021, 41(1): 43-47.

[17] 古天龙, 徐周波. 有序二叉树决策图及应用[M]. 北京: 科学出版社, 2009: 92-111.

[18] Duong C T, Hoang T D, Yin H, et al. Efficient streaming subgraph isomorphism with graph neural networks[J]. Proceedings of the VLDB Endowment, 2021, 14(5): 730-742.

编辑:张所滨