文章编号：2095-2163(2020)12-0094-05 中图分类号：TP301.6 文献标志码：A

一种高效的图编辑距离计算方法

陈梓扬，王 璿，周军锋，陈子阳
（东华大学 计算机科学与技术学院，上海 201620）

摘 要： 图编辑距离是用来衡量图相似性的一个重要方法，在很多领域都有应用。图编辑距离问题是 NP-Hard 问题，现有图编辑距离算法主要基于顶点映射搜索，由于搜索空间大，时间和空间效率较低。本文提出一个高效的基于 A* 的图编辑距离算法，针对图对称性引起的映射冗余问题，利用 Symmetry-Breaking 方法，通过减少扩展映射数量，提高算法的运行效率。最后，在真实数据集上进行实验，实验结果验证了其优化效果。

关键词： 图编辑距离；图相似性；图对称性

An Efficient Algorithm to the Graph Edit Distance Computation

CHEN Ziyang, WANG Xuan, ZHOU Junfeng, CHEN Ziyang
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】 Graph edit distance (GED) is an important measure to perform similarity-based analysis between two graphs and is widely used in many applications. Because of the NP-hardness, existing algorithms for computing GED have large time and space consumption. This paper proposes an efficient GED algorithm, which improves the efficiency of A*_GED by utilizing Symmetry-Breaking technique to reduce the number of vertex mappings to extend. Finally, the comparison is performed on real dataset, and the experimental results confirm the efficiency of the proposed algorithm.

【Key words】 Graph edit distance; Graph similarity; Graph symmetry

0 引 言

图可以用来表示对象之间复杂的关系，因此在很多领域有着大量应用。分析并管理图数据有很多基本问题需要解决，近年来针对这些问题有大量研究。在这些问题中，如何衡量两个图的相似性是很多应用的基础。例如，图的分类，图的聚类以及图的相似性查找等等。

近年来，研究者们提出了诸多的图相似性计算方法[1]，在这些方法中，基于图编辑距离（Graph Edit Distance）的方法受到了很多关注，因为其适用于多种类型的图。图编辑距离是指将一个图转变为另一个图所需要的最小操作数量，可以用来衡量两个图的相似程度。图编辑距离问题是一个 NP-Hard 问题[2]，时间和空间都有很大的开销。因此，图编辑距离算法需要解决复杂度过高带来的开销过大的问题。

现有的图编辑距离算法主要通过搜索顶点之间的映射来进行，搜索过程可以视为扩展搜索树的过程，搜索树的结点即顶点的映射。现有方法可以被分为两类：最佳优先搜索和深度优先搜索。最佳优先搜索基于经典的 A* 搜索算法（A*_GED），在搜索时优先扩展编辑代价，估计值最小的映射，搜索到的第一个完整映射对应的编辑代价就是图编辑距离。深度优先搜索（DF_GED）则优先扩展搜索树中层次更深的映射，直到遍历完所有完整映射，取其中最小的编辑代价作为图编辑距离。

A*_GED 会占用大量空间，而 DF_GED 在找到局部最优解时会产生大量回溯，消耗大量时间[3]，效率不如 A*_GED。针对 A*_GED 搜索空间过大的问题，本文利用 Symmetry-Breaking 方法，通过降低扩展映射数量，可以同时降低时间和空间开销。

1 相关工作

1.1 问题定义

定义 1 图 G 为一个三元组，即 $G=(V,E,l)$。其中，V 表示点的集合；$E=V\times V$ 表示边的集合；令 Σ 表示标签集合，$l: V\cup E\rightarrow\Sigma$ 表示图中顶点或边到对应标签的映射。

定义 2 编辑操作有 6 种，即点或边的删除、插

作者简介：陈梓扬（1996-），男，硕士研究生，主要研究方向：图的相似性查询、图编辑距离；王 璿（1977-），女，博士，副教授，主要研究方向：生物信息；周军锋（1977-），男，博士，教授，博士生导师，主要研究方向：大图数据的查询处理技术、推荐系统关键技术；陈子阳（1973-），男，博士，教授，博士生导师，上海立信会计金融学院特聘教授，计算机学会（CCF）会员，主要研究领域：数据库理论与技术。

通讯作者：周军锋 Email：zhoujf@dhu.edu.cn

收稿日期：2020-10-20

入和替换。假设存在点 u，v 和空点 ε，则点 u 的删除操作表示为 $(u \to \varepsilon)$，点 v 的插入操作表示为 $(\varepsilon \to v)$，点 u 替换为点 v 表示为 $(u \to v)$；边的 3 种操作的表示方法与此类似。

定义 3　如果图 q 和图 g 满足以下 4 个条件，则称 q 与 g 同构：

(a)存在一个从 $V(q)$ 到 $V(g)$ 的双射 f；

(b) $l(v) = l(f(v))$，$\forall v \in V(q)$；

(c) $(v,v') \in E(q)$，当且仅当 $(f(v),f(v')) \in E(g)$；

(d) $l(v,v') = l(f(v),f(v'))$，$\forall (v,v') \in E(q)$。

定义 4　编辑图 q，使得其于图 g 同构，把这一系列编辑操作的序列集合记为 k，e_i 表示第 i 步编辑操作，$k = (e_1,\cdots,e_i,\cdots,e_n)$，则称 k 为图 q 和 g 间的编辑路径。在图 q 和 g 间的所有编辑路径中，长度最短的路径成为最优编辑路径。

定义 5　在图 q 和 g 间的所有编辑路径中，长度最短的路径成为最优编辑路径。图编辑距离是最优编辑路径的长度，记作 $\delta(q,g)$。

直接使用定义 5 进行图编辑距离计算，需要枚举编辑路径，代价很大。目前常通过顶点映射定义图编辑距离[4]。不失一般性，规定 $|V(q)| = |V(g)|$。$V(q)$ 与 $V(g)$ 之间可以建立一一映射关系 f（以下简称映射）。

对于某一个完整映射 f，定义编辑代价 $\delta_f(q,g)$。

定义 6　遵循映射 f（即 $v \in V(q)$，$f(v) \in V(g)$）前提下，将图 q 转化使得其与图 g 同构所需的最小代价，记作 $\delta_f(q,g)$。

对于部分映射 f，定义编辑代价下界 $\underline{\delta}(f)$。

定义 7　对于所有由 f 扩展得到的完整映射，f 的编辑代价下界不超过其编辑代价的最小值，记作 $\underline{\delta}(f)$。

因此，可以给出基于映射的图编辑距离定义。

定义 8　图编辑距离可以用式(1)定义：

$$\delta(q,g) = \min_{f \in F(q,g)} \delta_f(q,g) \quad (1)$$

其中，$F(q,g)$ 表示所有从 $V(q)$ 到 $V(g)$ 的映射。因此，计算图编辑距离，可以通过搜索编辑代价值最小的顶点映射进行。

1.2　相关算法

已有不少工作基于搜索顶点映射进行图编辑距离计算。现有算法从空映射开始，逐个映射顶点搜索编辑代价最小的完整映射。由于图编辑距离问题是 NP-Hard 问题，搜索空间的大小与顶点数量呈指数关系。为了提高搜索效率，需要采取一些搜索策略。

1.2.1　最佳优先搜索

目前最佳优先搜索主要基于 A* 搜索算法[5]。A* 搜索算法维护一个搜索前缘，由一个优先队列 Q 组成，算法初始状态下，队列中只有一个空映射，接着开始迭代。每次迭代，A* 都会从 Q 中取出一个编辑代价下界最小的映射，并对其进行扩展，即计算子映射的编辑代价下界，并将其加入到队列中。A* 算法能够保证，第一个从队列中取出的完整映射，其编辑代价等于图编辑距离[3]。A*_GED 需要在优先队列中保存大量中间搜索状态，占用大量空间。

1.2.2 深度优先搜索

A*_GED 的问题是需要占用大量空间，近来有研究采用深度优先搜索来避免这一问题。深度优先搜索需要维护一个 $\delta(q,g)$ 的上界 $\bar{\delta}(q,g)$，是搜索过程中遇到的完整映射编辑代价的最小值。DF_GED 利用编辑代价下界来对搜索树进行剪枝[6-7]。如果遍历到的部分映射 f 满足 $\delta^{LS}(f) \geqslant \bar{\delta}(q,g)$，那么以这个映射为根结点的子树会被剪枝。否则，DF_GED会访问它的所有子映射，并对子映射进行扩展。虽然空间占用优于 A*_GED，DF_GED 通常会陷入局部最优解中，在搜索树的某个子树上进行大量的回溯，花费大量的时间，相比 A*_GED 效率很低[8]。本文常用符号见表 1。

表 1　常用符号

Tab. 1　Frequently used notations

符号	描述
q,g	目标图 q，查询图 g
$V(q),E(q)$	q 顶点数和边数
$\delta(q,g)$	q，g 之间的图编辑距离
$\bar{\delta}(q,g)$	$\delta(q,g)$ 的上界
f	从 $V(q)$ 到 $V(g)$ 的(部分)映射
$\delta_f(q,g)$	完整映射 f 对应的编辑代价
$q[f]$	q 中对应于映射 f 已映射部分的子图
$q\backslash f$	q 中除去 $q[f]$ 剩下的子图
$\underline{\delta}(q\backslash f,g\backslash f)$	将 $q\backslash f$ 编辑为 $g\backslash f$ 的编辑代价下界
$Y(S_1,S_2)$	$\max\{|S_1|,|S_2|\} - |S_1 \cap S_2|$
$\underline{\delta}(f)$	f 的编辑代价下界
$N(u)$	顶点 u 的邻接信息
$code(f)$	映射 f 等价类编码

2 A*_GED 与优化方法

2.1 A*_GED

A*_GED 算法根据映射 f 的编辑代价下界 $\delta(f)$ 启发式搜索,优先选择 $\delta(f)$ 最小的映射进行扩展。算法维护一个优先队列 Q,初始状态下,队列中只有一个空映射,接着开始迭代。每次迭代,A* 都会从 Q 中取出一个编辑代价下界 $\delta(f)$ 最小的映射 f,并对其扩展,即计算其子映射的编辑代价下界,并将子映射加入到队列中。A*_GED 算法能够保证第一个从队列中取出的完整映射,对应的编辑代价等于图编辑距离[3]。

例如,图 1 是查询图和目标图,图 2 是对其进行映射搜索产生的映射搜索树,π 表示顶点映射顺序,树结点表示映射,树结点旁的 $(u,\delta(f))$,表示将 v_i 映射到 u,形成的部分映射的编辑代价下界为 $\delta(f)$。如,$f_5=\{v_1\rightarrow u_1,v_2\rightarrow u_2,\}$,$\delta(f)=2$。在图 2 中,一开始队列 $Q=\{f_0\}$。第一次迭代,取出 f_0,加入其子映射,之后 $Q=\{f_1,f_2,f_3,f_4\}$。第二次迭代,从中取出编辑代价下界最小的 f_1,放入其子映射 $\{f_5,f_6,f_7\}$,使得 $Q=\{f_2,f_3,f_4,f_5,f_6,f_7\}$。第三次迭代,得到 $Q=\{f_2,f_3,f_4,f_6,f_7,f_8,f_9\}$,以此类推,直到取出的映射是完整映射。在图 2 的搜索树上运行 A* 算法,最终得到的完整映射是 f_{12},其编辑代价是 4,则图 q,g 的图编辑距离为 4。为了图示的简洁,映射没有全部画出。

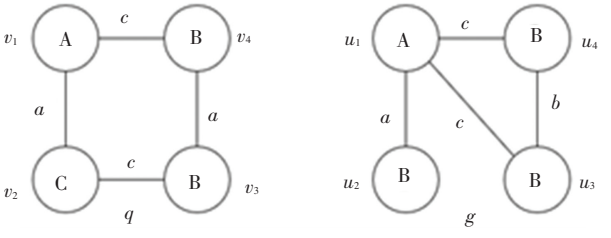

图 1 查询图 q 与目标图 g

Fig. 1 Query graph q and target graph g

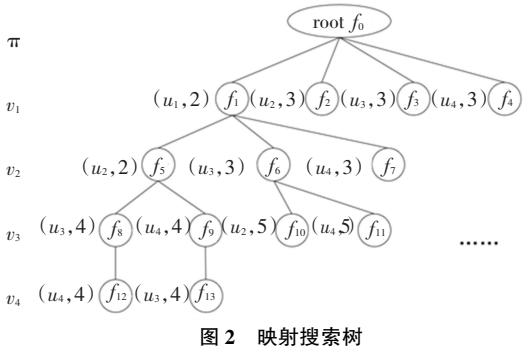

图 2 映射搜索树

Fig. 2 Vertex mapping search tree

A*_GED 算法存在扩展冗余映射的问题。由于图存在对称性,搜索树中很多映射是等价的。在搜索时,重复扩展这些相同的中间映射,会浪费大量的时间和空间。例如,图 1 中,顶点 u_3 和 u_4 事实上是等价的。扩展映射 $f_5=\{v_1\rightarrow u_1,v_2\rightarrow u_2\}$ 时,需要为 v_3 选择映射顶点,可能的顶点有 u_3 和 u_4,即子映射有 $f_8=\{v_1\rightarrow u_1,v_2\rightarrow u_2,v_3\rightarrow u_3\}$,$f_9=\{v_1\rightarrow u_1,v_2\rightarrow u_2,v_3\rightarrow u_4\}$。由于 u_3 和 u_4 顶点标签一致,邻接关系也一致,因此与 v_3 匹配产生编辑代价下界是相等的,$\delta(f_8)=\delta(f_9)=4$。不仅如此,以其为根的子树也是等价的。也就是说,在搜索树中,f_8 与 f_9 只需要保留其中一个。

实际应用中,图的对称性很普遍。比如有些化合物的分子式,内部存在大量等价的顶点。对其进行图编辑距离的计算,会因为对称性浪费大量的时间空间。

本文利用 Symmetry-Breaking 消除对称性引起的冗余映射,从而降低时空开销,提高计算效率。

2.2 Symmetry-Breaking

Symmetry-Breaking 是一个用来打破图的对称性的方法。Grochow 等曾把其应用在 motif 发现问题中[9];Chen 等提出的 GED 计算方法曾使用过类似的思想来消除 GED 计算过程中产生的重复中间映射,但消除对称性的程度不高[3]。当对称的顶点相互连接,例如图 2 中 u_3,u_4 顶点,该方法无法消除对称性。本文提出的方法能更好地消除对称性。

对于 $V(g)$ 中的每一个顶点 u,定义其邻接信息为 $N(u)=\{(v,l(u,v)):v\in V(g)\wedge(u,v)\in E(g)\}$。$N(u)/u'=\{(v,l(u,v)):v\in V(g)\wedge(u,v)\in E(g)\wedge v\neq u'\}$,表示 $N(u)$ 除去 v' 的邻接信息。

定义 9 等价顶点:对于 $u,v\in V(g)$,u 等价于 v,记作 $u\sim v$。$u\sim v$ 当且仅当 $l(u)=l(v)$ 并且 $N(u)/v=N(v)/u$。

例如 图 2 中,$N(u_3)/u_4=N(u_4)/u_3=\{(u_1,c)\}$。根据顶点之间的等价关系,可以将 $V(g)$ 划分为 λ_g 个等价类 V_g^1,$V_g^2\cdots V_g^{\lambda_g}$。如果 $u\in V_g^i$,就说 u 属于 i 类,记作 $\theta(u)=i$。例如,可以将图 2 中的 $V(g)$ 划分为 3 个等价类,$\theta(u_1)=1$,$\theta(u_2)=2$,$\theta(u_3)=\theta(u_4)=3$。

对于一个映射 $f=\cup_{l=1}^{|f|}\{v_{i_l}\rightarrow u_{j_l}\}$,可以根据等价类关系给出一个等价类编码。$code(f)=(\theta(u_{j_1}),\cdots,\theta(u_{j_{|f|}}))$。例如,$code(f_8)=code(f_9)=(1,2,3)$。可以证明,如果使用 LS 或 LSa 的编辑代价下界估计方法,对于等价类编码相同的两个映射,其编

辑代价下界是相等的，可以视为等价映射。

定理 1　若 $code(f_1) = code(f_2)$，则 $\delta(f_1) = \delta(f_2)$。

证明　为叙述清晰，证明以 LS 算法为例，LSa 算法同理。$\delta(f) = \delta_f(q[f], g[f]) + \underline{\delta}(q\backslash f, g\backslash f) = \delta_f(q[f], g[f]) + Y(L_V(q\backslash f), L_V(g\backslash f)) + Y(L_E(q\backslash f), L_E(g\backslash f))$ [8]。其中，$q[f]$ 和 $g[f]$ 分别为 q 和 g 的已映射部分，$q\backslash f$ 和 $g\backslash f$ 分别表示 q 和 g 的未映射部分。$L_V(q\backslash f)$ 表示 $q\backslash f$ 中包含的顶点标签集合，$L_V(g\backslash f)$ 同理。$L_E(q\backslash f)$ 表示 $q\backslash f$ 中包含的边标签集合，$L_E(g\backslash f)$ 同理。由 δ_f 的算法易得 $\delta_{f_1}(q[f_1], g[f_1]) = \delta_{f_2}(q[f_2], g[f_2])$。$Y(S_1, S_2) = \max\{|S_1|, |S_2|\} - |S_1 \cap S_2|$ 是集合 S_1, S_2 之间的编辑距离[8]。f_1 与 f_2 定义域相同，则 $L_V(q\backslash f_1) = L_V(q\backslash f_2)$，$L_E(q\backslash f_1) = L_E(q\backslash f_2)$。根据等价类的定义，显然 $L_V(g\backslash f_1) = L_V(g\backslash f_2)$，$L_E(g\backslash f_1) = L_E(g\backslash f_2)$。所以 $Y(L_V(q\backslash f_1), L_V(g\backslash f_1)) + Y(L_E(q\backslash f_1), L_E(g\backslash f_1)) = Y(L_V(q\backslash f_2), L_V(g\backslash f_2)) + Y(L_E(q\backslash f_2), L_E(g\backslash f_2))$ 综上，$\delta(f_1) = \delta(f_2)$。

在扩展子映射时，对于等价类编码相同的子映射，如果只保留其中一个，就可以避免大量重复的时间和空间消耗，定理 1 证明了该做法的正确性。算法 1 是消除等价顶点重复性的 A*_GED 算法。从第 9 行到第 12 行，在为 v_{i+1} 计算候选映射顶点集合时，去掉了等价类相同的点。

算法 1　A*_GED with Symmery-Breaking

输入：图 q, g

输出：q 与 g 的图编辑距离 $\delta(q,g)$

(1) generate mappingorder $\pi = (v_1, v_2, v_3, \cdots, v_{|V(q)|})$
(2) compute $\theta(u_i)$ for each $u_i \in V(g)$
(3) init priorityqueue Q, push emptynode $(\emptyset, 0, 0)$
(4) while $Q \neq$ do
(5)　pop $(f, i, \delta(f))$ from Q
(6)　/* full mapping found, return */
(7)　if $i = |V(q)|$ then return $\delta(f)$
(8)　/* compute candidates for v_{i+1} */
(9)　$C(v_{i+1}) \leftarrow \emptyset$
(10)　for each $u_i \in V(g\backslash f)$ do
(11)　if $\nexists\, u_j \in C(v_{i+1}), \theta(u_j) = \theta(u_i)$ then
(12)　add u_i to $C(v_{i+1})$
(13) extend $(Q, f, i+1, C(v_{i+1}))$
(14) Procedure extend $(Q, f, j, C(v_j))$
(15) for each $u_i \in C(v_j)$ do
(16)　compute $\delta(f \cup \{v_j \rightarrow u_i\})$
(17)　push $(f \cup \{v_j \rightarrow u_i\}, j, \delta(f \cup \{v_j \rightarrow u_i\}))$ into Q

分析 Symmetry-Breaking 的搜索空间，即搜索树的大小。记 $|V(q)| = |V(g)| = |V|$。搜索树可以按层划分，第 l 层的树结点个数记为 N_l。记搜索空间为 $S_R = \sum_{l=0}^{|V|} N_l$。对于 l 层的树结点，需要从 $V(g)$ 中选取 l 个顶点进行映射。记这 l 个顶点为 $B_g^l = \{u_{j_1} \cdots u_{j_l}\}$。用一个向量 $\boldsymbol{x} = [x_1 \cdots x_{\lambda_g}]$ 表示其分类情况，即 x_m 表示 B_g^l 中有 x_m 个顶点属于 V_g^m。显然，有下式

$$\sum_{m=1}^{\lambda_g} x_m = l, 0 \leqslant x_m \leqslant |V_g^m|, 1 \leqslant m \leqslant \lambda_g. \quad (2)$$

式(2)的一个解 $\boldsymbol{x}$，对应于一个独特的 B_g^l。记 Ψ_l 为式(4)所有解的集合。一个解 $\boldsymbol{x}$ 可以产生 $\frac{l!}{\prod_{m=1}^{\lambda_g} x_m!}$ 个等价类编码。例如，$\boldsymbol{x} = [2,1,0]$，可能产生的等价类编码为 $<1,1,2>$ $<1,2,1>$ $<2,1,1>$。每个等价类编码对应于一个从 B_q^l 到 B_g^l 独特的映射，即搜索树结点。那么，$N_l = \sum_{x \in \Psi_l} \frac{l!}{\prod_{m=1}^{\lambda_g} x_m!}$。

当 $l = |V|$ 时，显然式(4)有唯一解 $\boldsymbol{x} = [|V_g^1|, |V_g^2|, \cdots, |V_g^m|]$。因此 $N_{|V(g)|} = \frac{|V|!}{\prod_{m=1}^{\lambda_g} |V_g^m|!}$。因为 $N_0 = 1, N_1 \leqslant N_2 \leqslant \cdots \leqslant N_{|V|}$，所以，$S_R \leqslant |V| \frac{|V|!}{\prod_{m=1}^{\lambda_g} |V_g^m|!} + 1$，即 $S_R = O(\frac{|V||V|!}{\prod_{m=1}^{\lambda_g} |V_g^m|!})$。

3　实验分析

3.1　实验环境

实验所使用的硬件配置：

(1) CPU：Intel Xeon E5-1620 3.6GHz
(2) RAM：64GB
(3) 操作系统：Ubuntu 20.04 64 位

编程语言：C++

编译器：G++ 9.3.0

3.2　数据集

本文考察的图为无向图，带有顶点和边标签，化

合物的分子数据与本文所需要图数据近似。选取的实验数据来自 PubChem 数据集,是图相似性研究中的常用数据集。数据集中,图的顶点为原子,顶点标签为原子类型,边为原子之间的化学键,边标签为化学键类型。表 2 为 PubChem 的详细参数。

表 2 PubChem 数据集

Tab. 2 PubChem Dataset

数据集名称	平均 $\|V\|$	平均 $\|E\|$	$\|L_V\|$	$\|L_E\|$
PubChem	24	25.8	81	3

将数据集按照图的顶点个数进行分组,对于某一个整数 i, 顶点个数在 $[i-2,i+2]$ 区间内的图会被分入一组。比如对于 $i=5$, 顶点个数在区间 $[3,7]$ 之内的图会被分入一组。取 $i \in \{5,10,15,20,25,30\}$ 六组进行实验,每组 30 个查询,每个查询由一对查询图、目标图组成。

3.3 实验指标

为比较优化效果,以 AStar$^+$-LSa 为基准,对比本文提出的优化方法。实验获取 3 个指标,每个指标均取 30 个查询的平均值:

(1)图编辑距离的计算时间;

(2)扩展映射的数量,即从优先队列出队,需要进行扩展计算的映射数量;

(3)总映射数量,反应计算所需的空间。

3.4 实验结果与分析

实验得到的计算时间见表 3,中间扩展映射数量见表 4,总映射数量见表 5。

表 3 计算时间

Tab. 3 Running Time

i	AStar$^+$-Lsa(ms)	Symmetry-Breaking
5	0.054	0.051
10	0.79	0.64
15	306.64	214.03
20	8 729.98	5 871.22
25	38 437.40	16 413.71
30	29 363.70	22 802.19

表 4 扩展映射数量

Tab. 4 Number of expanded mappings

i	AStar$^+$-Lsa	Symmetry-Breaking
5	15.07	13.47
10	152.10	107.97
15	46 814.17	34 054.37
20	1 071 083.80	741 774.77
25	3 470 354.25	1 566 321.35
30	226 0961.87	1 784 663.27

表 5 总映射数量

Tab. 4 Number of enqueued mappings

i	AStar$^+$-Lsa	Symmetry-Breaking
5	24.47	18.50
10	489.90	309.20
15	237 033.00	157 511.43
20	6 642 829.53	4 281 243.93
25	32 377 533.70	12 211 274.40
30	26 026 531.93	19 058 130.80

对比 AStar$^+$-LSa 与 Symmetry-Breaking,可以很明显的看出。Symmetry-Breaking 具有不错的时空优化效果。以 $i=25$ 的图为例,空间上,总映射数量优化约 62%。时间上,扩展映射数量降低 54%,计算时间约降低 57%,说明计算时间上的优化主要来自扩展映射数量的减少,这符合预期。对于 $i=5$ 的小图而言,优化效果不如大图明显,总映射数量约优化 25%,时间优化 5%,这是由于小图的扩展映射计算量不大,且对称顶点较少,优化空间不大。对于 $i \geqslant 10$ 的图,空间上优化效果在 36%~60%,时间上的优化效果在 20%~57%。总体而言,空间占用平均降低了 41%,时间消耗平均降低了 37%。因此,实验验证了 Symmetry-Breaking 的高效性。

4 结束语

针对 A*_GED 算法在计算图编辑距离时效率较低的问题,本文使用了 Symmetry-Breaking 方法,定义了顶点之间的等价关系,通过降低等价的冗余映射数量,同时优化了时间和空间效率,实验结果表明,空间占用平均降低了 41%,时间消耗平均降低了 37%,具有不错的优化效果。

参考文献

[1] 徐周波, 张鹃, 宁黎华,等. 图编辑距离概述 [J]. 计算机科学, 2018, 45(4): 11-8.

[2] ZENG Z, TUNG A K, WANG J, et al. Comparing stars: On approximating graph edit distance [J]. Proceedings of the VLDB Endowment, 2009, 2(1): 25-36.

[3] CHEN X, HUO H, HUAN J, et al. Fast Computation of Graph Edit Distance [J]. arXiv preprint arXiv:170910305, 2017.

[4] BLUMENTHAL D B, BORIA N, GAMPER J, et al. Comparing heuristics for graph edit distance computation [J]. The VLDB Journal, 2020, 29(1): 419-458.

[5] RIESEN K, FANKHAUSER S, BUNKE H. Speeding up Graph Edit Distance Computation with a Bipartite Heuristic[C]// Mining and Learning with Graphs, MLG 2007, Firence, Italy, August 1-3, 2007, Proceedings. DBLP, 2007.

[6] BLUMENTHAL D B, GAMPER J. Exact Computation of Graph Edit Distance for Uniform and Non-uniform Metric Edit Costs [C]// International Workshop on Graph-Based Representations in Pattern Recognition. Springer, Cham, 2017.

[7] ABU-AISHEH Z, RAVEAUX R, RAMEL J Y, et al. An exact graph edit distance algorithm for solving pattern recognition problems [M]. 2015.

[8] CHANG L, FENG X, LIN X, et al. Efficient graph edit distance computation and verification via anchor-aware lower bound estimation [J]. arXiv preprint arXiv:170906810, 2017,

[9] GROCHOW J A, KELLIS M. Network motif discovery using subgraph enumeration and symmetry-breaking: proceedings of the Annual International in Computational Molecular Biology, International Conference, Recomb, Oakland, Ca, Usa, April. Springer. Berlin, Heidelberg, 2007.