Ullmann 子图同构算法

一、 子图同构问题一般解法描述

如下，图 MA，MB 分别表示两个图 A 和 B 的对应的边矩阵，其中 MA[i][j]=1 表示顶点 vi 和顶点 vj 有边。

	1	2	3
1	0	1	1
2	1	0	0
3	1	0	0

MA

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

MB

如下 M’矩阵表示映射 g 从 A 到 B 的映射矩阵，M’[i][j]=1 表示矩阵 A 中的顶点 i 和矩阵 B 中的顶点 j 之间的一个映射。在最基本的情况下，我们认为任意两点之间都可以映射，那么我们的工作就是需要枚举所有的映射情况，进一步判断当前映射是否满足子图同构的条件，来获得同构子图的一个映射。

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

M’

而 Ullmann 算法的工作，就是涉及如何快速（剪枝）枚举所有的映射情况，进而寻找所有的同构子图。

二、 映射 g 满足性质基础

如果图 A 关于映射 g，与图 B 子图同构，令：

$$MC = M'(M' \cdot MB)^T \tag{1}$$

则：

$$\forall i \forall j : (MA[i][j] = 1) \Rightarrow (MC[i][j] = 1) \tag{2}$$

所以，对于一个枚举的映射情况，我们进行两次矩阵运算，得到矩阵 MC 之

后，进行上述公式（2）的判断，满足情况则视为找到了一个同构子图；否则当前情况不可行，继续枚举。

另外，子图同构映射 g 的矩阵 M’需满足以下性质：

（1） M’[i][j]=1 表示 A 中的第 i 个顶点对应 G 中的第 j 个顶点；

（2） M’每行仅有一个 1；

（3） M’的每列中 1 的个数至多有一个。

而 Ullmann 算法的提出就是用来寻找所有的矩阵 M’（和第三部分最后同一个说法，这里再次说明一下）。

三、 Ullmann 算法流程

（1） 建立矩阵 M

若 A 中的第 i 个顶点与 B 中的第 j 个顶点有相同的标签、且第矩阵 A 中的第 i 个顶点的度小于等于矩阵 B 中的第 j 个顶点的度，则令 M[i][j]=1。

（2） 从矩阵 M 中枚举矩阵 M’

对于每个枚举的矩阵 M’，每行有且仅有一个 1，每列最多有一个 1.

（3） 对于每一个枚举的矩阵 M’，进行上述（2）式的判断。

（4） 迭代以上步骤，列出所有可能的矩阵 M’

对于每个映射矩阵 M’，我们可以输出一个对应的同构子图。

四、 优化

前提：在进行枚举之前，尽可能地减少 M 矩阵中的 1 的数目，就可以减少之后枚举的数量，进而提升算法整体效率。

依据：通过顶点之间的连边特征，来减少 M 矩阵中 1 的数目（之前仅仅是利用顶点 label 是否相同以及顶点度数关系来判断矩阵 A 中的顶点 i 与矩阵 B 中的顶点 j 是否可映射。

改进：对于图 A 中的顶点 i 与图 B 中的顶点 j，在图 A 中所有与 i 有关联的点 x，需要与图 B 中顶点 j 的某些点对应。即满足：

$$(\forall_{1\le x\le N} x)\quad (MA[i][x]=1)\Rightarrow \exists_{1\le y\le M} y\quad (M[x][y]\bullet MB[y][j]=1)$$

注：此过程可迭代进行，至当前轮没有 M[i][j]由 1 改为 0 停止。

五、 实验结果：

因为算法复杂度十分高，在已有查询上运行出结果时间太长，下面展示对某

些查询集合进行查询的部分结果（人工验证查询结果正确）：

对于 Q4 查询部分结果：

映射 g1（查询图 id=0 -> 数据库中图 id=0）：

原图顶点	0	1	2	3	4
映射图顶点	0	2	4	6	10

映射 g2（查询图 id=0 -> 数据库中图 id=0）：

原图顶点	0	1	2	3	4
映射图顶点	1	0	2	4	6

对于 Q8 查询部分结果：

映射 g1（查询图 id=0 -> 数据库中图 id=17）：

原图顶点	0	1	2	3	4	5	6	7	8
映射图顶点	2	0	3	1	5	7	10	8	6

映射 g2（查询图 id=0 -> 数据库中图 id=17）：

原图顶点	0	1	2	3	4	5	6	7	8
映射图顶点	2	0	3	1	6	8	10	7	5

注 1：输入数据按照无向图处理。

注 2：多考虑边之间 label 是否相同。

参考资料：

1. http://blog.csdn.net/chichoxian/article/details/52746232
2. http://blog.csdn.net/chichoxian/article/details/52748456
3. http://www.cnblogs.com/huadongw/p/4154295.html