基于信息熵的子图匹配算法*

孟凡荣，张　青，闫秋艳
（中国矿业大学 计算机科学与技术学院，江苏 徐州 221116）

摘　要：子图查询是指输入一个图数据库和查询子图，输出图数据库中包含查询子图的图集合，它广泛应用于社会网、生物网和信息网的查询应用中。目前的子图查询算法大多采用静态消耗测算模式，此类测算模式在图中点数和连接边数呈指数分布时，会在少数节点上花费较多时间遍历其邻节点，导致查询算法效率低下。根据信息熵在信息度量中的作用，将条件信息熵作为启发式匹配的依据，提出了基于信息熵的子图匹配算法。实验表明，基于信息熵的子图匹配算法具有更高的查询效率，且在指数分布的数据集上效果更明显。

关键词：图数据；信息熵；子图匹配
中图分类号：TP391；TP301.6　　**文献标志码**：A　　**文章编号**：1001-3695(2012)11-4035-03
doi：10.3969/j.issn.1001-3695.2012.11.008

Information entropy based algorithm for efficient subgraph matching

MENG Fan-rong, ZHANG Qing, YAN Qiu-yan
(*School of Computer Science & Technology, China University of Mining & Technology, Xuzhou Jiangsu* 221116, *China*)

Abstract: Subgraph matching query refers to input a query graph and a data graph, and output the graph which contains all the subgraphs of the data graph. Subgraph query is widely used in social network, biological network and the query application of the information network. Current work on subgraph matching queries used static cost models which could be ineffective due to long-tailed degree distributions, for it would spend more time in traversing the adjacent node. According to the information entropy on the basis of information measure, using the conditional information entropy as the basic for the heuristic matching subgraph matching algorithm, this paper proposed an information entropy based algorithm for efficient subgraph matching. Experiments show that the proposed method has a higher efficiency of inquires. And in the long-tailed degree distributions of dataset, the effect is more apparent.

Key words: graph; information entropy; subgraph matching query

0　引言

随着 Facebook 等社交网络类应用的兴起，针对海量图数据的查询和处理日益广泛。同时，由于使用了 RDF(resource description framework)作为网络数据的存储标准，在信息网络和语义 Web 中引入带标签的边表示节点之间的关系，形成结构更为复杂、信息量更为丰富的图数据结构。因此，针对大规模、复杂图数据的子图查询具有重要且广泛的应用前景。

子图同构是子图查询的核心问题，是一个 NP 完全问题[1]，目前还没有提出过多项式时间复杂度的算法，各种已有算法的时间复杂度都是指数级的。已有子图同构算法均基于回溯搜索方法，尝试把查询图在数据图上逐步扩展，如果扩展成功就继续向前扩展，如果失败则回退到前一步继续尝试其他的搜索路径。比较经典的算法有 Subdue 算法[2]、Ullman 算法[3]、Nauty 算法[4]、VF-2 算法[5]等。Shang 等人提出了 quick-SI 算法[6]，并在 quick-SI 算法中提出了一种基于特征的子图测试方法，具有很高的效率，其本质也是经典回溯搜索算法的变种。以上算法中的回溯算法均采用静态消耗测算模式[7]，即在扩展过程中需要事先统计邻节点。

在 RDF 领域，主要关注于数据的存储结构和查询策略[8]。在查询策略方面，现有的工作主要针对子图匹配算法本身和索引结构进行优化。在匹配算法上多采用基于统计的启发式算法[9~11]。上面所述算法均采用静态消耗测算模式。由于统计邻节点需要耗费大量时间，尤其是对现实中大量存在的指数分布的图数据，会在少数节点上花费较多时间遍历其邻节点，导致查询算法效率低下。本文根据信息熵在信息度量中的作用，将条件信息熵作为启发式匹配的依据，提出了基于信息熵的子图匹配算法，将条件信息熵作为启发式匹配的依据，以减少邻节点的匹配次数，提高子图查询的效率。实验表明，基于信息熵的子图匹配算法具有更高的查询效率，且在指数分布的数据集上效果更明显。

1　相关概念

1.1　图数据的相关定义

本节给出图数据相关的一些基本概念[7]。

定义 1　图数据。一个图数据 G 定义为一个三元组 $G=\langle V,E,L\rangle$。其中，V 表示图中顶点的集合；L 表示标签的集合；$E\subseteq V\times L\times V$ 表示标签边的集合。

收稿日期：2012-04-09；**修回日期**：2012-05-24　　**基金项目**：国家自然科学基金资助项目(71003096,61003169)；高等学校博士学科点专项科研基金资助项目(20110095110010,20100095110003)

作者简介：孟凡荣(1962-)，女，辽宁沈阳人，教授，博导，博士，主要研究方向为数据库技术、数据挖掘（mengfr62@163.com）；张青(1989-)，男，硕士研究生，主要研究方向为数据挖掘、信息安全；闫秋艳(1978-)，女，副教授，博士，主要研究方向为不确定数据挖掘、流数据挖掘.

定义 2 查询子图。查询子图 Q 是一个集合对 (V_Q, E_Q) 其中 $V_Q \subseteq V \cup \mathrm{VAR}$ 且 $E_Q \subseteq V_Q \times L_Q \times V_Q$。在查询子图 Q 中用 VAR_Q 表示可变点的集合。

定义 3 置换映射。θ 是关于图 G 的查询子图 Q 的映射 $\theta: \mathrm{VAR}_Q \rightarrow V$。

定义 4 查询结果。置换映射 θ 是关于图 G 的查询子图 Q 的一个结果。关于图 G 的子图查询 Q 的结果集表示为 $AS(Q, G)$。

1.2 信息熵

熵理论是由美国数学家 Shannon 提出的，是测定信息的不确定性。根据信源的信息选择不定度的测定，确定信息表征信源的不确定度[12]。

熵的公式为

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) \tag{1}$$

联合熵、熵、条件熵之间的关系为

$$H(XY) = H(X) + H(Y|X) = H(Y) + H(X|Y) \tag{2}$$

2 基于信息熵的子图匹配算法

2.1 引入信息熵

在数据图中，每一个事件的发生与否都具有一定的信息量。在事件未发生前，事物所具有的平均信息量称之为信息熵[12]。在巨大的社会网中，由于每个节点之间的关系是复杂且不可知的，加之大部分点之间的关系非常小，所以研究单个节点的信息熵是没有意义的。本文中认为不相邻节点之间是相互独立的，所以在此讨论的是相邻节点之间的条件熵。

在图数据中，用 x 表示当前点，即所需要求信息熵的点；$x_1\ x_2 \cdots\ x_i$ 表示 x 的邻节点中符合条件的点。根据式(2)可以推得

$$H(Z|XY) = H(X|Z) + H(Y|Z) + H(Z) - H(X) - H(Y) \tag{3}$$

同理

$$\begin{aligned} H(x|x_1x_2\cdots x_i) &= H(xx_1x_2\cdots x_i) - H(x_1x_2\cdots x_i) = \\ &H(x_1x_2\cdots x_i|x) + H(x) - H(x_1x_2\cdots x_i) = \\ &H(x_1|x) + H(x_2|x) + \cdots + H(x_i|x) + H(x) - H(x_1x_2\cdots x_i) = \\ &H(x_1|x) + H(x_2|x) + \cdots + H(x_i|x) + H(x) - \\ &H(x_1) - H(x_2) - \cdots - H(x_i) \end{aligned} \tag{4}$$

在图数据 G 中任一个点的信息熵为

$$H(x) = -\frac{1}{x}\log_2\left(\frac{1}{x}\right) = \frac{\log_2 x}{x}$$

由于图数据 G 中的节点非常多，可认为 $x \rightarrow \infty$，即 $H(x) \rightarrow 0$。由此可得

$$\begin{aligned} &H(x|x_1x_2\cdots x_i) = \\ &H(x_1|x) + H(x_2|x) + \cdots + H(x_i|x) \end{aligned} \tag{5}$$

可见在图数据中，选择点 x 的条件熵等于此点以其各个符合条件的邻节点 $x_1\ x_2 \cdots\ x_i$ 为条件的熵的和。在下一步匹配的节点选择上，必然先选择那些信息熵大的点，因为它们提供了更多的信息，有利于下一步的选择。

2.2 算法描述

基于信息熵，本文提出了 entropyMatch 算法。具体算法如下。

```
1) 函数 entropyMatch
input:查询子图 Q。
A←∅
for all z∈V_Q//选出确定查询点
  if z∈VAR_Q then R_z←∅
  else R_z←{z}
P←{(z,R_z)|z∈V_Q}
//存放确定查询点及其潜在匹配
entropy(P,Q,G)
//计算 P 中确定查询点的条件熵
selectVertex(Q,∅,P)
2) 函数 entropy
input:可能匹配集 P,查询子图 Q,图 G。
H←0
for all z∈P
  for all edges e=(z,l,v)∈V_Q
    N←retrieveNeighbors(G,m,l)
//遍历出匹配点 m 的邻节点的潜在匹配的集合 N
    if N=null then H←H
/*如果查询点的邻节点太多以至于 retrieveNeighbors 返回值为 null,H 的值不变*/
    else H←H+calcEmtropy(N)
//计算查询点的条件熵,calcEmtropy 计算单个邻节点的条件熵
H_z←{H}
3) 函数 selectVertex
input:查询子图 Q,部分置换映射 θ,可能匹配集 P。
for all H∈H_z
  if H_m=max(H_z)
//找到熵值最大的查询点
substituteVertex(Q,θ,P,m)
A←A∪{θ}
//将潜在映射集 θ 加入到结果集 A 中
4) 函数 substituteVertex
input:查询子图 Q,部分置换映射 θ,可能匹配集 P,查询点 w,可能匹配 m。
retriveVertex(G,m)
//在图 G 中找到可能匹配点 m
if w∈VAR then θ'←θ∈{w→m}
Q'←Q;
for all z∈V_Q  R_z'←R_z
for all edges e=(w,l,v)∈E_Q'
    N←retrieveNeighbors(G,m,l)
    if N=0 then R_z'←R_z-m;
continue
//如果点 m 没有符合条件的邻节点则从 R_z'中删除点 m
if N≠null then
    if R_v'=∅ then R_v'←N
    else R_v'←R_v'∩N
//查询点是否已被初始化,若选过则求交集
remove e from Q'
//将边 e 从查询子图 Q 中去掉
    if R_z'=∅ break
    else P'←{(z,R_z')|z∈V_Q}
entropy(P',Q',G)
selectVertex(Q',θ',P')
```

每个函数的具体含义如下：

a) entropyMatch 函数。先对确定的查询点进行广度优先算法。在遍历的过程中，通过函数 entropy 求出各个已确定点的条件熵。其中 (z, R_z) 表示关于查询子图中点 z 的可能匹配；R_z 用于存储 z 的潜在匹配。当 z 在查询子图中是常量时，R_z 只包含 z；当 z 在查询子图中是变量时，R_z 为空；在 z 被初始化后，R_z 为 z 的一系列可能的匹配(其中一些值在算法运行过程中可能被删除)。在关于图 G 上的查询子图 Q(Q 中至少包含一个确定点)的函数 entropyMatch 运行结束后，返回值为 $AS(Q, G)$。

b) entropy 函数。对 R_z 中的每个可能的匹配点进行匹配，对每个节点的边进行匹配，通过调用函数 retrieveNeighbors 遍历出匹配点 m 的邻节点中符合匹配的集合 N。在函数 retrieveNeighbors 中，当点的信息熵小于图的各个点的平均信息熵时，retrieveNeighbors 的返回值为 null，因为在遍历过程中计算熵值

太小的点是不希望发生的。

c)selectVertex 函数。从信息熵的集合 H_z 中选出熵值最大的点 m,并调用算法 substituteVertex;再将符合要求的部分置换映射 θ 加入到可选点集合 A 中。

d)substituteVertex 函数。调用 retriveVertex 函数在图 G 中对 m 点进行索引,并对部分置换映射 θ、查询子图 Q、z 的潜在匹配集 R_z 进行更新;然后查找点 m 的邻节点的潜在匹配点并将它们添加到 R_z' 中,同时将边 e 从 Q' 中删除;最后,更新可能匹配集 P'中的值并调用算法 entropy 计算 P'中可选点的信息熵。

3　实验

本文实验在数据集 UCI-credit 和数据集 Webdata[13] 上对 entropyMatch 算法进行评价,与 Subdue[2] 算法进行比较,对算法的时间效率进行测试。其中数据集 Webdata 指数分布性质更加明显。算法使用 C 语言实现,用带有-O2 优化选项的 gcc 编译;实验环境为 RedHat Linux 8.0 操作系、单核 CPU、1GB 内存。

本文针对上述两个数据集对 entropyMatch 进行测试,评估查询包括六个子图查询匹配,子图中边的个数为 1 ~7 不等。本文设计查询子图的顺序是:首先确定查询结构,然后确定固定点,最后从数据集中随机选取标签初始化查询子图。子图结构的定义如表 1、2 所示。在本文中,用 Subdue 算法与 entropyMatch 算法进行比较,比较的结果如图 1、2 所示。图表中的数据都是多次运行程序取平均值所得。

表 1　在 UCI-credit 数据集中图和查询图的参数

UCI-credit	图 G	查询图 Q_1	查询图 Q_2	查询图 Q_3
点数 V	14 700	6	7	8
边数 E	14 000	5	6	7
不同的标签数 L	79	9	11	13

表 2　在 Webdata 数据集中图和查询图的参数

Webdata	图 G	查询图 Q_4	查询图 Q_5	查询图 Q_6
点数 V	10 238	2	4	6
边数 E	10 625	1	3	5
不同的标签数 L	2 036	3	5	6

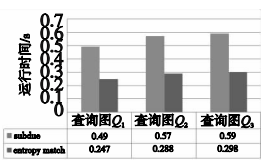

图1　UCI-credit数据集上两种算法查询时间比较

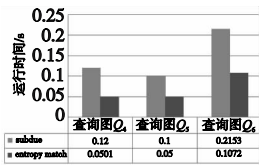

图2　Webdata数据集上两种算法查询时间比较

从图 1、2 中可以看出,entropyMatch 算法在不同复杂度的图和不同复杂度的查询子图下都具有更高的时间效率。图 3、4 表明,在相同的时间下,entropyMatch 算法具有更高的准确度。图 5 表示在各个数据集上 entropyMatch 算法比 Subdue 算法效率提高的百分比。从图中可以看出,算法在数据集 Webdata 上提高的效率均比在数据集 UCI-credit 上高得多。由此可得 entropyMatch 算法在指数分布的数据集上优势更加明显。

4　结束语

单个大图下的子图匹配问题是一个 NP 完全问题。与之相关的一些算法由于采用静态消耗测算模式因而效率低下。本文针对这个问题,依据信息熵在信息度量中的作用,将条件信息熵作为启发式匹配的依据,提出了基于信息熵的子图匹配算法 entropyMatch。实验中分别在 UCI 公测数据集和数据集 Webdata 上对算法性能进行比较。在每个数据集上均选取三个查询子图,表明在不同复杂度的子图下算法的效率。实验表明,entropyMatch 算法具有更高的效率,且在限定时间内有更高的准确度,尤其是在现实中大量存在的指数分布的数据集上。

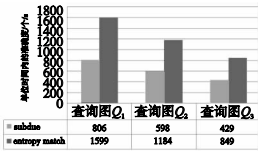

图3　单位时间内匹配结果数比较（UCI-credit数据集）

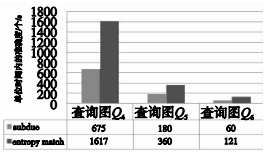

图4　单位时间内匹配结果数比较（Webdata数据集）

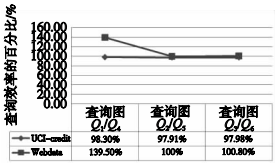

图5　查询效率比较

参考文献:

[1] GAREY M R, JOLMSON D S. Computers and intactbility:a guide to the theory of NP-completeness[M]. New York:W H Freeman &Co, 1990.

[2] KETKAR S, HOLDER B, COOK J. Subdue:compression-based frequent pattern discovery[C]//Proc of ACM KDD Workshop on Open-source Data Mining. 2005:71-76.

[3] ULLMAN J R. An algorithm for subgraph isomorphism[J]. Journal of the ACM, 1976, 23(1):31-42.

[4] McKAY B D. Practical graph isomorphism[J]. Congressus Numerantium, 1981:45-87.

[5] L CORDELLA, P. FOGGIA, C. SANSONE, M. VENTO. A (sub) graph isomorphism algorithm for matching Iarge graphs[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2004, 26(10):1367-1372.

[6] SHANG Hai-chuan, ZHANG Y, LIN X, *et al*. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism[C]//Proc of International Conference on Very Large Data Bases. 2008:364-375.

[7] BROCHELER M, PUGLIESE A, SUBRAHMANIAN V S. A budget-based algorithm for efficient subgraph matching on huge networks[C]//Proc of the 27th International Conference on Data Engineering. Washington DC:IEEE Computer Society, 2011:94-99.

[8] MAHMOUDINASAB H, SAKR S. An experimental evaluation of relational rdf storage and querying techniques[C]//Proc of the 15th International Conference on Database Systemfor Advanced Applications. Berlin:Springer-Verlag, 2010:215-226.

[9] CHENG J, KE Y, NG W. Efficient query processing on graph databases[J]. ACM Trans on Database System, 2009:34(1).

[10] GIUGNO R, SHASHA D. GraphGrep:a fast and universal method for querying graphs[C]//Proc of the 16th International Conference on Pattem Recognition. 2002:112-115.

[11] KE Yi-ping, CHENG J, YU J X. Querying large graph databases[C]//Proc of the 15th International Conference on Database Systems for Advanced Applications. 2010:487-488.

[12] SHANNON C E. Mathematical theory of communication[J]. ACM SIGMOBILE Mobile Computing & Communications Review, 1948(Vol. 27):79-423.

[13] SUBDUE[EB/OL]. http://ailab. wsu. edu/subdue/.