# 挖掘拓展相关理论

## 拓展使用到的相关性质和定义

包含关系：若两个满足以下条件：

1.

2.为的一个子图，且存在匹配满足。

我们称。

该包含关系具有传递性。

相等关系：若两个满足以下条件：

1.

2.

我们称

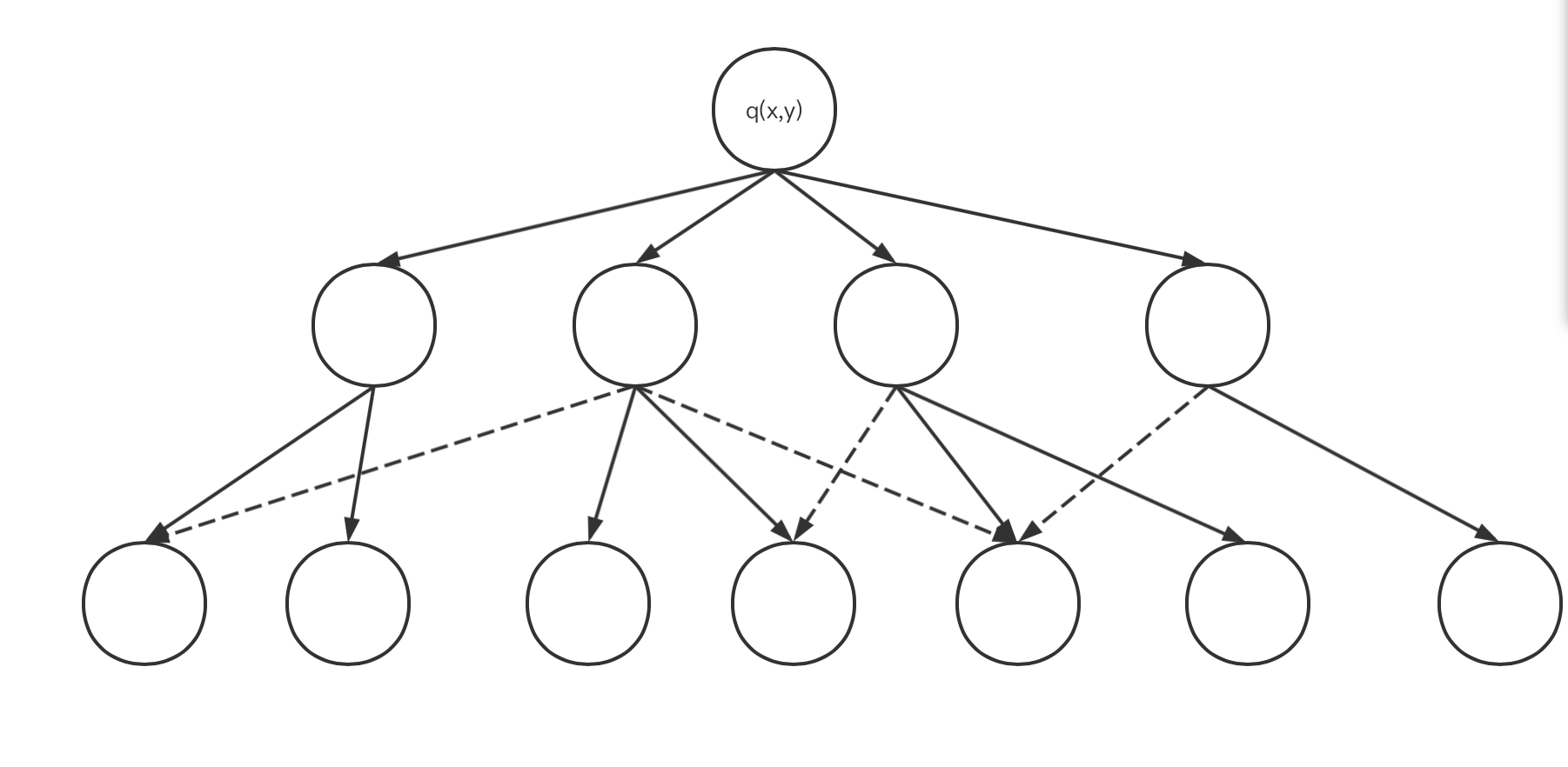
性质1:若两个满足,则

,

性质1具有传递性：对于三个，若满足性质1，满足性质1，则满足性质1。

## 拓展图

拓展图即为在挖掘过程中生成的一个表现形式，一个拓展图可以表现成如下形式：



在拓展图中，每个节点表示的是一个,其中第层的节点表示边数为的的集合，且不存在两个节点表示的相等。其中根节点表示的是的。

在拓展图中，若两个节点存在一条实的有向边，则表示代表的是由代表的加了一条边得到的。若两个节点存在一条虚的有向边，则表示代表的可以通过添加一条边的方式得到一个与代表的相等的。

## 与拓展树的区别

1.拓展图比拓展树具有更少的节点。每个节点就代表了一次子图同构的计算，因此采用拓展图模型进行的拓展相对于用拓展树模型进行拓展会减少子图同构的计算次数。

2.拓展图相对于拓展树而言，每个节点具有更多的祖先关系，可以利用祖先信息进行剪枝从而可以不通过子图同构的方式就可以确定该是否满足的条件。

## 拓展树的相关定理

结合拓展树的结构以及性质1，我们可以得到以下几个定理。

定理1:考虑拓展图上的一个节点,若其存在一个祖先节点满足条件,则。

定理2:考虑拓展图上的一个节点,若存在一个节点满足以下条件中的任意一个：

,且

,且

则

定理3: 考虑拓展图上的一个节点,若其在拓展图上的父节点集合为，则

其中，定理1和定理3是性质1及其传递性在拓展图上的表现形式，定理2则是针对（即不存在匹配）这个特殊情况，因为此时节点可能不是节点的祖先，但是其可能存在包含关系。

通过定理1和定理2，可以不通过在数据图上匹配的方式来确认该是否需要保留。定理3则是通过祖先关系确定了该的可能的，从而减少了子图同构的匹配次数。

## 利用定理来进行剪枝

通过定理1和定理2，我们可以通过以下算法来确定一个是否需要计算和。

算法的输入参数主要为三个：表示要判断的,表示的是目前的挖掘过程中通过大图匹配发现的或者的集合，表示的是的集合。整个算法分为两个部分：第一部分是根据定理2来进行检查，若该满足定理2，则不需要进行计算。第二部分是根据定理1来进行检查，若找到一个的是在拓展树上的祖先节点，则不需要进行计算。

|  |  |
| --- | --- |
|  | |
|  | : |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |

## 拓展图的生成方式

拓展图可以通过以下方式生成：

对于第层的集合中的每个,通过加边得到新的,并查询当前已经生成的中是否存在与相等的,若存在，则添加一条的虚有向边，否则添加一条的实有向边并将加入到生成的集合中。

通过上述方式生成拓展图的复杂度是，其中是该拓展图的边数，表示的是在的集合中查找与相同的的复杂度。

通过复杂度分析可以得出，构建该拓展图的性能瓶颈主要在于查找是否存在相等的操作。

## 

上一部分提到在拓展图的构建过程中，其构建过程中最费时是过程，一般的操作器思路是逐个比较并判断是否相同，其涉及到子图同构且逐个扫描的复杂度是线性的，在实际的挖掘过程中是难以接受的。因此，提高的性能是十分重要的。

借鉴了频繁子图挖掘中对于这样情况的处理——对每个构建一个编码，则将变为编码的比较。同时也借鉴了的，即用过程来对进行编码。提出了适用于的,计作。

相比于的有以下差别：

的只能用于判断“该是否已经出现”而不能做到“该与哪个相同”，而则满足了全序性，因此可以将其插入到二叉平衡树中方便查找，同时也可以实现“查询与哪个相同”的功能。

## 相关定义

序列：给定一个,其序列是一个的三元组列表，其中满足以下条件：

1. 其序列反映在进行的过程。

下标：给定一个以及其序列，将出现过的节点按照出现次序用0,1,2…..进行标记。

: 给定一个,其是一个的五元组列表，其中表示和的下标，同时该满足以下条件：

1. 其序列反映在进行的过程。

同时，我们将的结构计作。

的比较：若两个,满足以下条件中的任意一个：

我们称

的比较：给定两个，若满足以下条件:

存在一个位置，使得。

我们称。

根据以上定义，我们可以得到以下性质：

性质4:满足全序性质。

证明：将每个映射成一个整数值，则一个可视为一个字符集为的字符串，其中表示的是映射的整数值的最大值。因为字符串的比较方式是满足全序关系的，因此自然也满足全序关系。

最小：对于一个,我们将其多个中最小的作为该的最小。

显然，我们可以得到以下性质：

性质5:若两个的最小相同，则这两个相等。

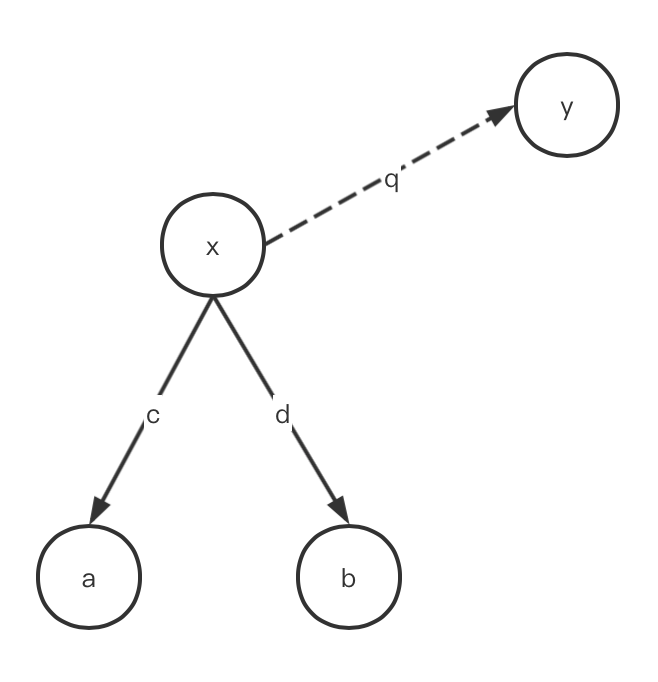
下面是一个最小的具体例子。

考虑下面的,则其为：

(0,1,x,y,q),(0,2,x,a,c),(0,3,x,b,d)

(0,1,x,y,q),(0,2,x,b,d),(0,3,x,a,c)

而其最小为(0,1,x,y,q),(0,2,x,a,c),(0,3,x,b,d)。



## 相关算法

根据,我们可以得到的一个具体算法

|  |  |
| --- | --- |
|  | |
|  | :, |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
|  |  |
| 7 |  |
| 8 |  |
| 10 |  |

在过程中，除了输入的外，同时还维护一个,表示到的映射。则其过程则是先构建好的,再将其在上进行查询，若查询有结果，则通过定理3更新和，否则就将其插入到中。

## 相关算法

基于的定义，可以知道求一个的最小的复杂度为，其中为的边数。因此，本文提出了一个基于贪心方式的构建最小的方法，其算法的核心思想在于在的过程中先将边按照某种方式排序，再根据顺序遍历边。具体策略如下：

DFS的过程中，优先访问出边再访问入边。访问出边/入边时，对对应的出边/入边进行排序。出边排序规则：

若排序的是出边，每条边的形式写成(src\_script,dst\_script,src\_label,edge\_label,dst\_label)

选取dst\_script最小的边，对于未被访问的点,script统一计作INF

若所有边的script相同（即终点都未被访问），则选取edge\_label最小的边

若存在多条edge\_label最小的边，则选择dst\_label最小的边

若依然存在多条相同的边，则比较dst的指出的节点数量

若依然存在多条相同的边，则比较dst的指入的节点数量

若依然存在多条相同的边，则比较dst的指出的边数量

若依然存在多条相同的边，则比较dst的指入的边数量

依然相同，则选取最左边的边