DMine-Pro总结

# 概述

是挖掘算法的相关改进。主要目的是解决算法运行效率低下，无法完成边数较大的挖掘的问题。

## 研究现状

2015年，本组提出了图模式关联规则,并提出了的挖掘算法，随后以为基础，衍生出图函数依赖,图关联规则等新的规则。同时提出了一些新规则的挖掘算法。

但是在实际运用上，目前的挖掘算法均无法满足实际需求。主要存在以下问题：

1.挖掘算法的效率是十分低下的，虽然理论复杂度得到证明，但是其复杂度较高，很多时候无法在可观时间下进行挖掘，从而以挖掘为基础的相关图规则挖掘算法也存在同样的问题。

2.挖掘算法由于其策略相对保守，无法挖掘到边数较大的。若边数较少，则表示的语义有限。

基于这两个实际问题，目前以为基础的相关图规则挖掘算法都只是存在理论上可以运用的可能，无法在实际场景下运用，只能是理论产物。

## 贡献

针对以上两个问题，我们提出了改进的挖掘算法，该算法主要针对以下部分进行改进：

1.根据生成的特点提出了一套拓展的相关理论，用于减少挖掘过程中无用的个数以及提高去重的效率。

2.结合挖掘中出现的现象采用启发式搜索对搜索空间进行了部分舍弃以提高效率，同时保证了能挖掘出大部分期望的结果。

实验证明，相比于算法在运行效率上至少提升了两个数量级，同时也能通过挖掘出边数较大且具有现实意义的。不仅如此，中使用的优化经过适当修改即可用于以为基础而拓展出来的图规则挖掘算法，具有一定程度的泛用性。

# 相关理论与技术

## 

## 的和

## 的包含与相等关系

**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。

显然，的包含关系是具有传递性的。同时，根据的包含关系，我们可以得到以下性质：

**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。

**相等关系**：若两个满足以下条件：

1.

2.

我们称

根据的包含关系，

## 挖掘算法

的基础框架如下所示。

|  |  |
| --- | --- |
|  | |
|  | : |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |

其中，的输入为挖掘的数据图，需要预测的边，对于的的下界，最后保留的数量和的最大边数。

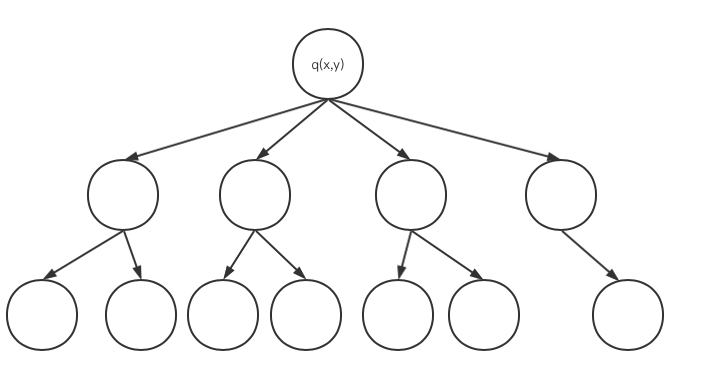
首先会初始化相关辅助的数据结构（算法的第一行），其中表示的是最后的的保存结果，表示的是当前的的最大边数，表示的是当前目前为止生成的全部的结果，表示的是上一轮保留进入下一轮拓展的。

算法的主要流程为算法的2-11行。该算法采用的是一轮一轮加边的增长模式，即通过上一轮的保留结果，将内的每个都增加一条新边（算法第4行），生成新的结果，计作。生成之后，由于会出现多个相同的,因此需要对生成的进行去重（算法第5行）。去重之后，就需要计算生成的的以及，同时将的去掉（算法第6行）。计算完和之后，就是将符合条件的插入到中，同时判断该是否可以进入下一轮拓展（算法第7-11行）。

# 挖掘拓展相关理论

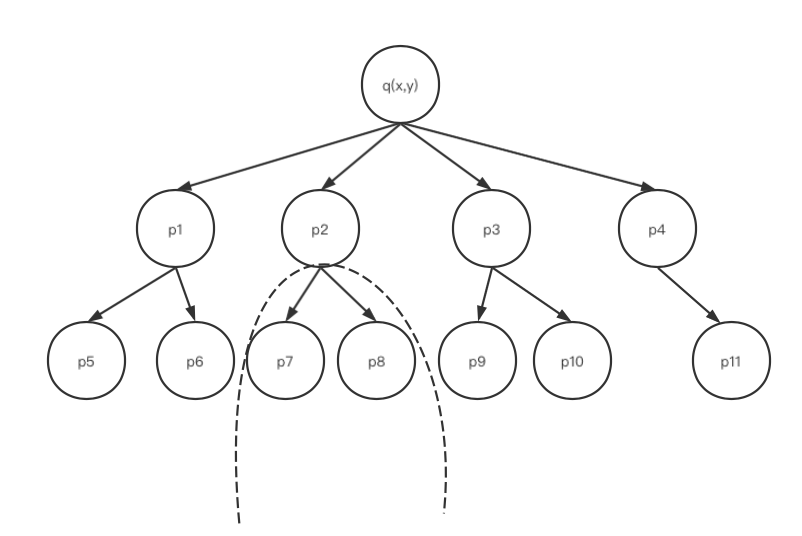
## 拓展树

的拓展过程可以用下图的拓展树来表示。在拓展树中，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。其中根节点表示的是的。在拓展树中，若两个节点存在一条有向边，则表示代表的是由代表的加了一条边得到的。



## 拓展图

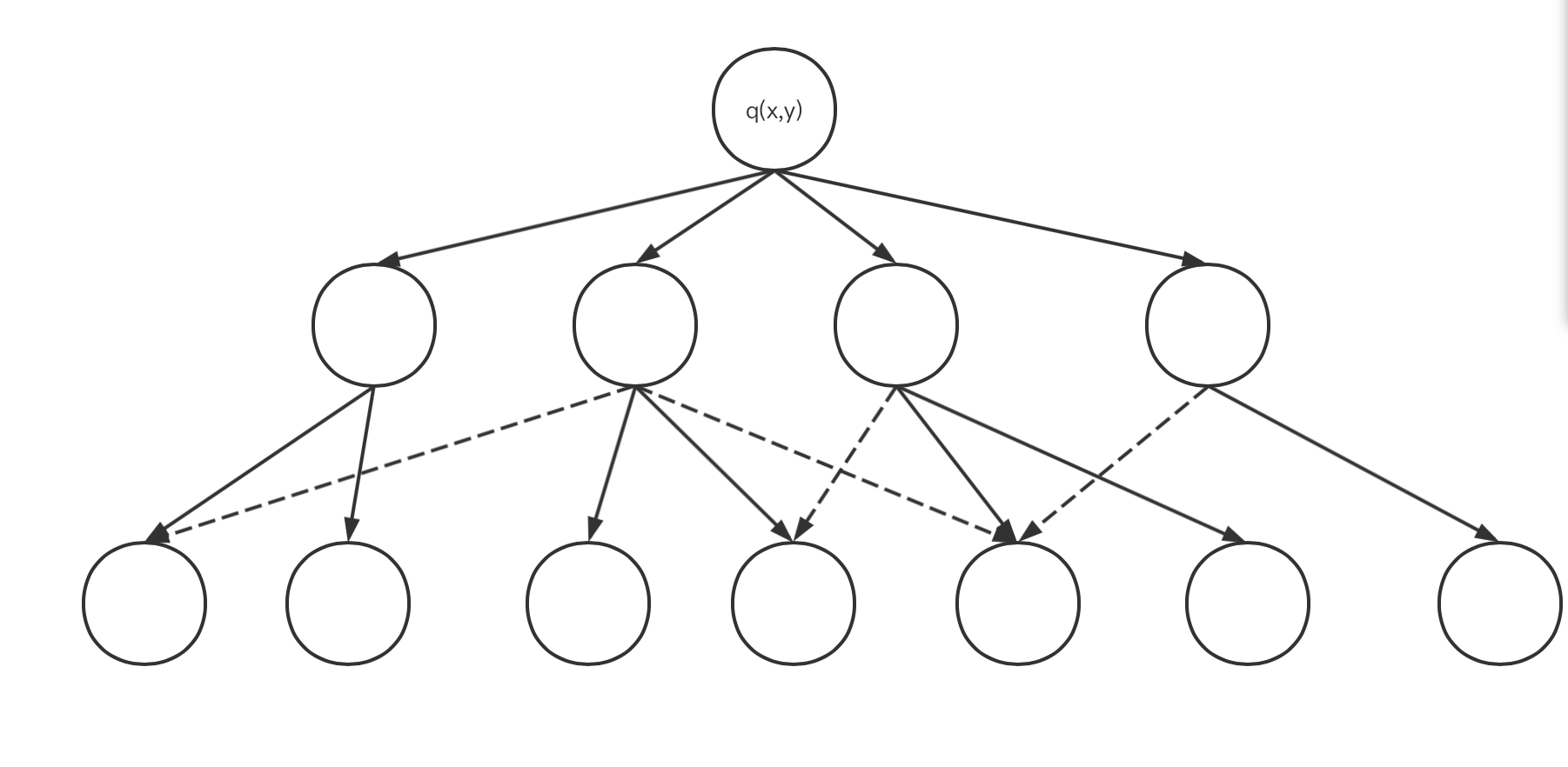
用拓展树来表示挖掘算法中的生成过程虽然简单直接，但是由于缺失了祖先信息，所以会让挖掘过程中产生了很多无用的计算。如下图所示。



在给定阈值的情况下，计算得出,因此，均会被剪枝。但是实际上，跟是相等的，而在拓展树上，是需要进行一次子图同构的计算才能得出。相当于这一次的计算是无用的。

会发生上图所示的现象的主要原因在于拓展树提供的信息是比较少的：一个边数为的最多存在个边数为的满足。而在拓展树上只标记了一个，造成祖先信息的缺失，因此无法进行有效的剪枝。

针对以上的情况，我们可以将拓展树稍作修改，添加适当的祖先信息，得到拓展图，如下所示。



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。

拓展图和拓展树存在以下区别：

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。

## 拓展图的相关定理

结合拓展图的结构以及性质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。

通过定理1和定理2，我们可以通过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

考虑到在挖掘过程中，的拓展是通过加边的方式生成的，因此直接构建拓展图可能有点难度，我们可以通过先构建拓展树，再将拓展树转化为拓展图，其转化方法如下所示：

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。

## 编码

在拓展树转化为拓展图的过程中，最费时的操作是过程，一般的操作其思路是逐个比较并判断是否相同，其涉及到子图同构且需要线性扫描一遍所有，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。

我们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。因此我们提出编码系统的相关定义。

**编码系统**：一个编码系统是一个的映射，且满足以下条件：

1.需要满足全序关系。

2.若两个的相同，则他们一定是相等的。

其中，第一个条件是为了满足能对所有的满足严格的偏序关系，从而可以将其插入到类似平衡二叉树的结构内方便查找。第二个条件是该编码设计的初衷——用编码相等的条件来代替子图同构。

在编码的设计上，我们借鉴了的思想，即用过程来对进行编码。提出了适用于的,计作。

## 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。

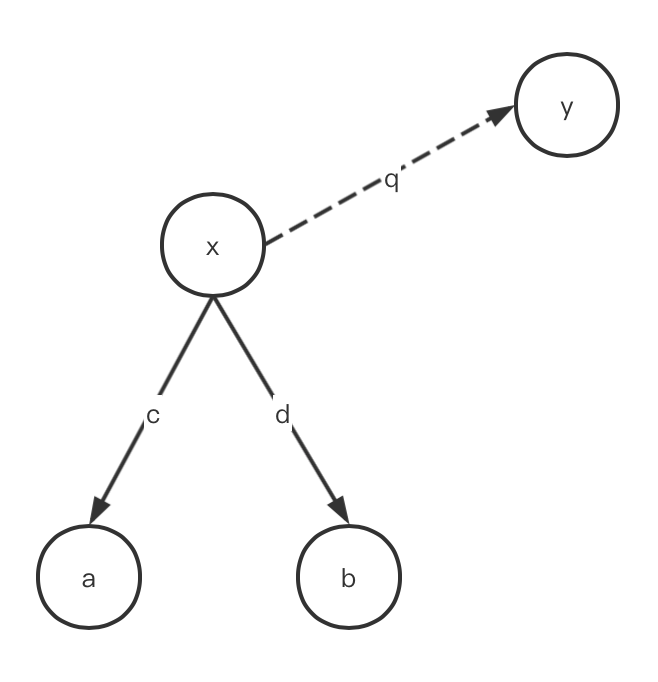
**证明**：将每个映射成一个整数值，则一个可视为一个字符集为的字符串，其中表示的是映射的整数值的最大值。因为字符串的比较方式是满足全序关系的，因此自然也满足全序关系。

**最小**：对于一个,我们将其多个中最小的作为该的最小。

下面是一个最小的具体例子。

考虑下面的,则其为：

而其最小为。



根据,我们可以得到的一个具体算法，其算法如下所示。

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。

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，我们提出了一个基于贪心方式的构建最小的方法，其算法如下所示。该算法与一般的的区别在于在对于每个点，会把该点的出边和入边按照某一种策略进行排序，之后按照排序结果进行逐个访问。在该算法中，我们采用的排序策略如下所示：

先选取script最小的边，若边的script相同，则edge\_label小的边在前面，若存在多条edge\_label最小的边，则选择dst\_label最小的边。

|  |  |
| --- | --- |
|  | |
|  | :, |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |

|  |  |
| --- | --- |
|  | |
|  | :, |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |

# 筛选

## 动机

在实际的挖掘过程中，会出现以下情况：

1.挖掘过程中很多的都是“类似”的。

2.目前的剪枝仅有“针对supp阈值的剪枝”，实际需求下，这个supp的阈值不一定设置的很高，所以很多时候该剪枝的效果是很差的。

3.在挖掘过程中，低conf的结果是占据大多数的。

例子：在联通数据图的挖掘过程中，若设置supp\_r=10,则会在edge=7时生成近千万级别数量的。若设置supp\_r=200,在edge=5时，生成的90%的的conf都在0.1-0.3之间。

## 解决思路

针对以上的几个问题，采用的主要解决思路是：

1.在生成时，尽量少生成“效果差的”

2.对于多个“相似”的，仅保留一个即可。

提出这样的思路主要是基于以下这两个假设：

1.如果两个是“相似”的，那么他们拓展出来的结果绝大部分也都是相似的。

2.如果一个的conf很低，那么他拓展得到的大部分的conf也都是比较低的。

假设1表示，对于“相似”的,考虑到大部分情况下他们拓展出的结果也是相似的，因此只需要保留一个拓展即可。假设2表示，在实际的挖掘过程中，最后得到的好的结果一般都是由效果还可以的拓展得来，因此我们可以在挖掘过程中适当舍弃一些低conf的结果。

## 相关定义

**相似**：给定一个小数，如果两个GPAR 满足以下条件：

则我们称是相似的。

根据相似的定义，我们可以得到如下性质：

**性质5**: 给定一个小数，如果两个GPAR ，则

**证明**：若 ，则, ，因此

若，则, ,因此

证明部分同理

该性质主要的用途是减少避免计算两个数据差别很大的是否相似，减少计算时间。

**筛选问题**：其具体定义为：

输入：待筛选的集合,小数和整数

输出：满足以下条件的集合:

1.

2.不存在两个相似的满足

3.最大

该筛选问题主要强调的是留下的不相似即可，相比于类似聚类的办法而言，不强调“哪些是相似的”，只强调“结果不相似即可”。同时，为了满足假设2，用指标作为保留下来的的质量评估。

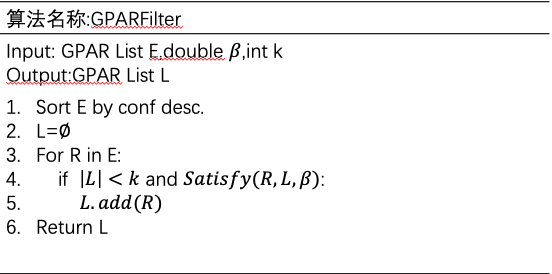
通过筛选问题我们可以得出其算法复杂度的结论。

**结论1**：筛选问题是一个问题。

**证明**：将中的每个视为一个节点，且节点权值为其，若两个相似则在这两个对应的节点加一条边，那么求筛选问题的最优解可规约为这个图上求最大带权独立集。因为求解一般图的带权最大独立集是问题，因此该问题也是一个问题。

## 近似算法

考虑到筛选问题是一个问题，若求精确解，其运行时间是无法接受的。因此采用贪心的想法来求一个近似解。其算法如下所示

**

该算法的核心思想在于先对中的根据conf从大到小排序，并逐个判断当前的是否可以加入到中，若可以则加入，不可以则跳过。

# 实验结果