# 公共结构以及分组查询(parallel Or distributed)

**定义1：**

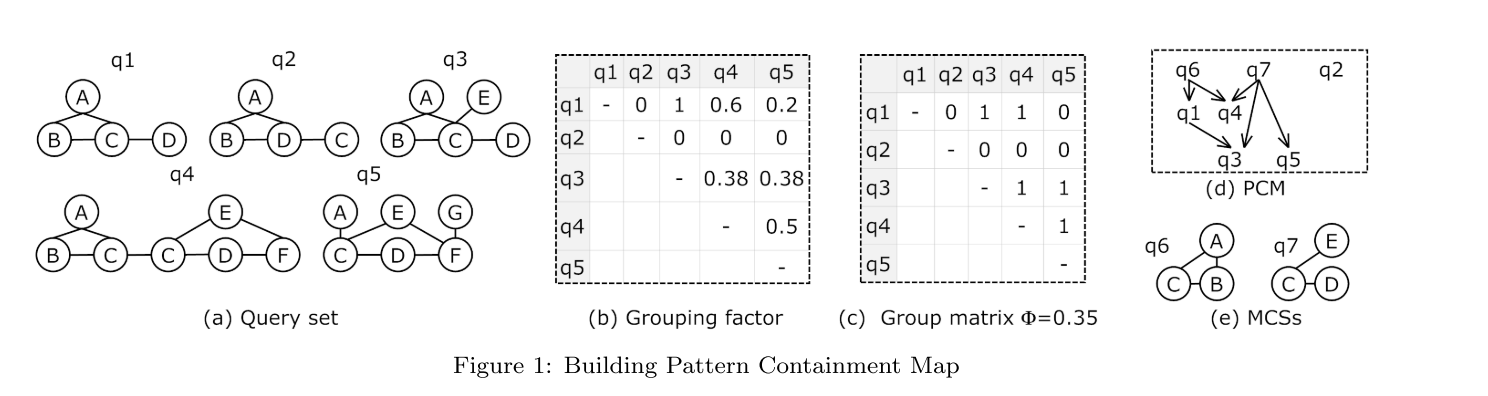
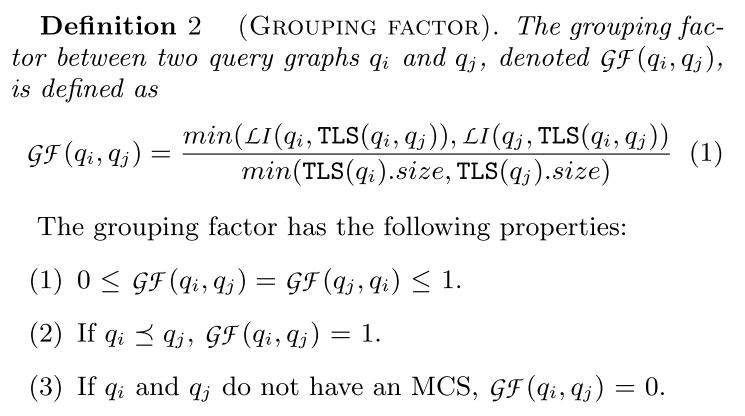
**(Tri-Vertex Label Sequence)** (Tri-Vertex Label Sequence). Given a pair of connected edges (vi, vj) and (vj, vk) of a super dependency tree TS, assuming L(vi) ≤ L(vk), we call the label sequence L(vi)- L(vj)-L(vk) a Tri-Vertex Label Sequence (TLS), and (vi, vj, vk) an instance of the TLS in TS. 

Figure 1. Super dependency tree set

Figure 1. TLS (TS3, TS4) ={(A-B-C), (A-C-B), (B-A-C), (D- C-E)}. We use L1(TS3, TLS(TSi, TSj)) to denote the number of instances in the largest instance subTree of TS3 corresponding to the TLSs in TLS(TSi, TSj). For example, for TS3, and TS4 in Figure 1, L1(TS3, TLS(TS3, TS4)) =4 and L1(TS4, TLS(TS3, TS4))=3。

之后根据如下定义计算grouping factor并根据阈值形成Fiuger1 c。



最终形成Figure 1中的d。

在这里PCM中的每一个箭头都代表两个TS之间存在公共结构。

# 3.分组执行

从这个地方开始不同于论文中的工作，我的设想是到这一步开始，将PCM中的超句法依存树进行分组。

即便是单机的系统，其并行度一般也不为1。也就是说，即便是只能在单机上执行的程序我们仍然可以根据硬件条件（CPU数量与核数），写出多线程或者多进程代码，因此程序可以并行执行。例如，对同一份RDF数据，同时执行SPARQL查询操作是没有问题的。

因此这引出一个新的问题。如何将查询进行分组，使其对应硬件的并行度。例如执行SPARQL查询时可能有这样的问题。有n个查询（不考虑公共结构），电脑的并行度是m，那么应该如何将这n个查询尽可能平均的分为m组非常重要。当然是尽可能的平均分最好。于是这个问题一定会引出下面这个问题，估计查询代价。

## 3.1 查询代价(时间)

假设有估计查询代价的函数cost(TS)，通过这个函数我们可以估计一个由句法依存树查询TS生成的SPARQL查询Q的代价。当然最好有两个代价函数:

(1)

(2)

其中公式1可以给出TS生成的SPARQL查询Q的代价，公式2可以给出由 生成的Qi和Qj公共查询Qc的代价。于是如果我们设计了多查询优化，公共查询只执行一次，于是就会有如下公式：

（3）

拥有了公式3，就可以计算在有公共结构的情况下，一组超句法分析树生成的SPARQL查询的代价。

目前已经有不少研究工作给出了使用机器学习的方法来预测SPARQL查询的代价的方法，例如14年的论文< A Machine Learning Approach to SPARQL Query Performance Prediction >以及18年WWW的一篇论文都给出了预测SPARQL查询代价的方法。

相比较于单纯的SPARQL查询给出的特征信息，事实上句法分析树中给出的特征信息更多。因此希望可以找到一个比较中肯的方法（机器学习算法）来实现预测由超句法分析树生成的SPARQL查询的代价。

## 3.2 PCM中句法分析树分组

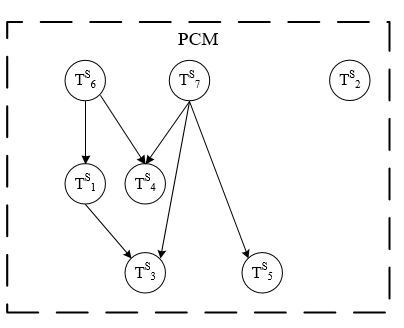


Figure 2 PCM

如图2所示，假设超句法分析树之间可以形成这样的连接关系。

**定义：**

**入度**：指向PCM中某一结点TSi的边数量。

**出度**：PCM中某一结点TSi指向其他结点的边数量。

**度**：入度+出度

不难看到，PCM中的所有结点可以分为以下三种情况

* 度为0
* 度为1
* 度>=2

分组的时候，对于度为0和1的结点都很好解决。度为0 的结点例如TS2不和任何其他结点有公共结构，因此自己就是一组。度为1的结点，例如TS5就一定是和指向它的结点TS4绑定在一组。

而针对度大于等于2的结点TSi就比较复杂。TSi可以与所有指向它的结点组合在一组，也可以与它指向的结点组成一组。为了更好的负载均衡。设计如下启发式的算法进行分组。

**算法如下：**

**输入**：PCM ，电脑并行度N

**输出**：将PCM中的所有超句法分析树分为N组

算法分为三种情况：

***PCM中度为1的结点数量>=N***

1. 挑出所有度为1的结点，分为N组。
2. 将所有与度为1的结点相连的结点加入对应组
3. 计算每组的查询代价
4. 将度为大于等于2的结点尝试加入有边与其相连的其他结点所属分组，（要加入查询代价最小的一组）
5. 最后分配度为0的结点。尽可能保持不同组之间代价基本相似。

**PCM中度为1的结点的数量>N/2且<N**

1. 按照度为1的结点的数量m分为m组
2. 将所有与度为1的结点相连的结点加入对应组

由于实际上服务器的并行度为N，而m<N，因此还剩下N-m分组可以进行计算。

1. 因此此步选择将度为0的结点平均分入这N-m个分组中。（但是要保证这N-m个分组中，最大的那一组代价不能超过那m个分组中的最大代价，如果开始超出，则将该超句法分析树分配至前m个分组中代价最小的一组）
2. 处理度大于等于2的结点。每次都选择当前m组中代价最小的那一组加入。（未必要是与该结点连的结点所属的组）

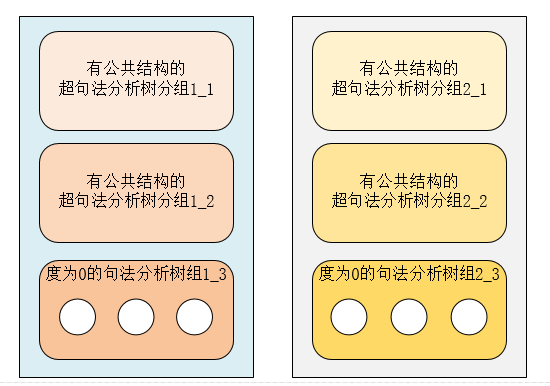
**PCM中度为1的结点数量小于N/2时**

未解决。

N/2是一个设定的参数，可以是N/2,也可以是N/3等等

## 组查询执行顺序问题

经过上面对PCM中句法分析树的分组。给出如下一个例子：



假设计算机的并行度为2，并生成了如上图的两个分组。在超句法分析树生成SPARQL查询Q的部分是没有问题的。这只涉及到计算。计算过程中对存储的压力依赖比较小。

但是当所有的超句法分析树都生成为SPARQL之后，要执行SPARQL查询问题就来了。上图中每个组对内存的耗费情况是不同的。 例如每一组从开始执行到执行完毕这段时间内耗费内存情况如下：

|  |  |  |
| --- | --- | --- |
| 执行顺序 | 组1 | 组2 |
| 1 | 500MB | 500MB |
| 2 | 100MB | 200MB |
| 3 | 3GB | 2GB |

如果只是单纯的要按照上表的执行顺序来执行。如果计算机的内存只有4GB，那么很显然。当执行第三步的时候麻烦就大了。由于内存受限。无法运行。计算机崩溃。但这本来是可以避免的。

**因此在此步应该有两步重要内容需要做。**

1. 如何使用机器学习算法来估测sparql查询占用内存的大小（可以使用每个triple pattern对应的数据所占存储空间大小之和）
2. 设计合理的組执行顺序算法。