# 摘要：

在mapreduce中基于left-deep-join的TwinTwigJoin，使用TwinTwig作为基础join单元，我们证明了在 ER随机图模型中，合适的假设下，TwinTwig join是最优的。我们也设计了一些算法来解决这个问题。 设计了3个optimization 策略用来提高算法效率。最后，我们讨论了为什么我们的方法能够被适用于 power-low random-graph模型。执行实验。

动机：

1. 子图迭代计算复杂，np-hard
2. 造成大量中间结果。尺寸很大。

现有的基于mapreduce的，edge-based和multiway join 两种方式。

Edge-based join:一轮join一个边，低效。 Multi-way join：每一轮到要有整个图，很复杂。

TwinTwig-join: trade-off edge-based和star-based join。首先证明了left-deep-join下最优。

1. 通过TwinTwig的结构获得最优的 执行计划。3.许多优化策略。

# 贡献：

1. left-deep-join。从edge-based join到multiple edge join
2. 提出TwinTwig算法
3. 针对上述算法提出优化策略
4. 拓展到power-law random graph的理论证明
5. 实验

# 2.问题定义

子图定义 匹配定义….

图存储：分布式存储，邻接链表。

假设：

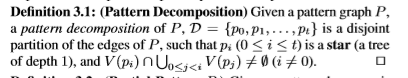
A1:数据图是ER随机图

A2:算法遵从left-deep-join框架，right join的参数是star

A3,数据图是稀疏的

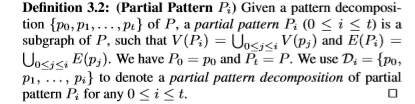
# 算法框架、

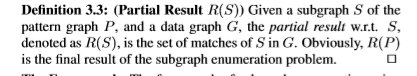
查询图分解：



每个子模式是 edge的disjoint partition。

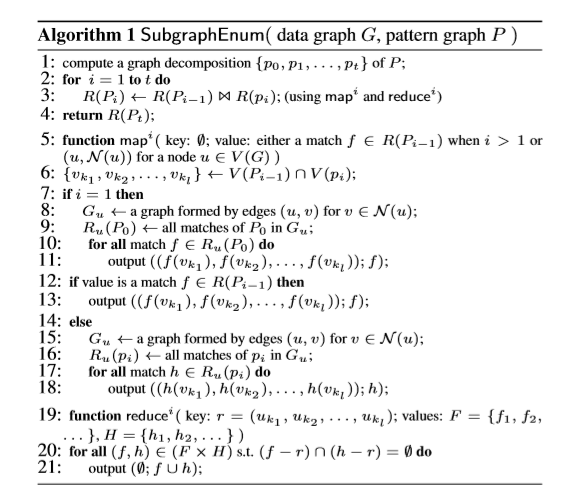
Partial Pattern P:





查询图P分解。从前往后的组合Pattern P。结果R(S)

算法框架：



第一行分解成执行计划。之后的t轮使用mapreduce join。

算法第六行：计算需要join的key。

1）如果i=1。我们需要计算R(P0)。7-11算出这样的匹配然后输出。

2）如果输入是个f，那么我们直接输出这样的匹配，

3）计算pi的匹配，输出。

Reduce:按照key分组，也就是说和这些

使用left-deep join能够提供较优的解。

相比bushy join保存的中间结果更少

我们保存的中间结果只和之前的那个有关，bushy需要很多轮。

计算最优join计划的代价更低。

# EXISTING SOLUTION

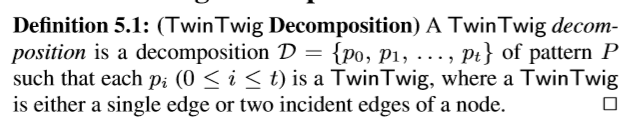
EdgeJoin：按照边分解，然后join。次数太多。

Starjoin:对于节点的度数大的，很有问题。

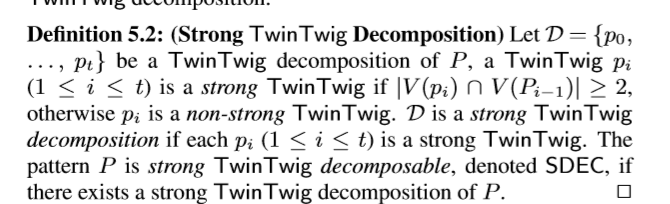
MultiwayJoin:一轮mapreduce。但是需要边的复制，特别当查询图变大时候，也会变大。进行了举例子分析。

# A new approach

## 5.1TwinTwig 分解



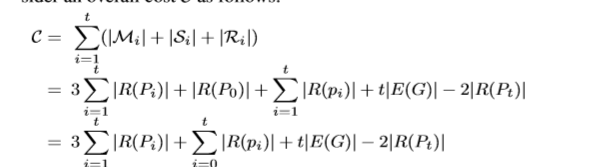
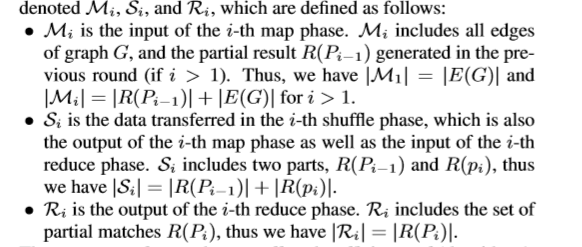
相比较star-join，避免了很多顶点的情况。相比较edge-join，更少轮。

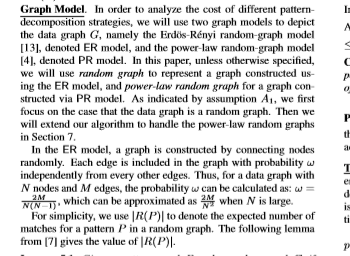


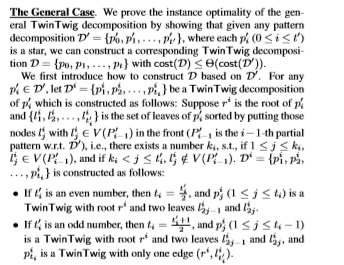
强的分解，下一个和上一个的交集顶点数大于等于2.

## Cost Analysis

定义了costmodel:



图模型：

如何从star-分解 转换成TwinTwigjoin

## Optimal Decomposition by A\*

动态优化

* 1. SymmetryBreaking

1. OPTIMIZATION STRATEGIES

6.1 Order-awareCostReduction

6.2 WorkloadSkewReduction

6.3 EarlyFiltering

# 7. HANDLINGPOWER-LAWGRAPHS

仍然给了一个强假设，不符合真实网络。