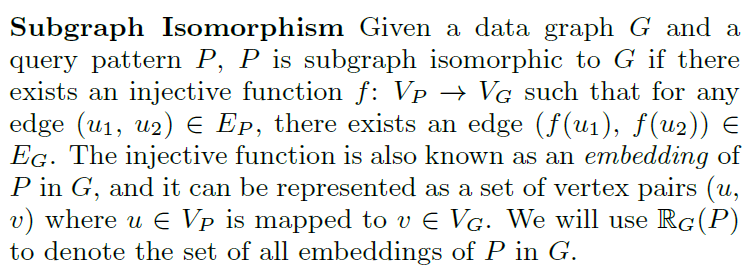
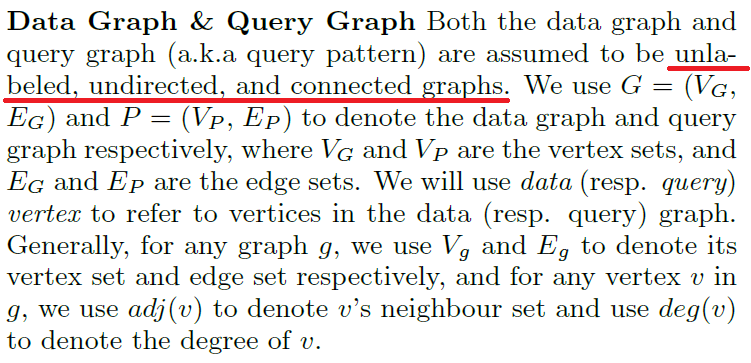
第一部分：背景与挑战



(什么是子图同构问题呢？假如给定了数据图G和查询图P，如果存在一个映射函数f，使得对于任意的(u1,u2)∈Ep,存在一条边(f(u1),f(u2)∈Eg，那么就称为P是G的一个子图同构。这个映射函数也称为P在G中的一个映射。实际上子图匹配问题就是找到所有的这种映射关系。



（子图迭代讨论的问题通常是无标签的，无向的联通图，在论文中通过用Vg和Eg表示数据图的顶点和边，Vp和Ep表示查询图的顶点和边。那么什么是子图迭代问题呢）

(Subgraph Enumeration) Given a pattern graph P and a data graph G, subgraph enumeration is to enumerate all subgraphs g of G such that g is isomorphic to P.

（子图迭代问实际上就是 给定了查询图P和数据图G，找到数据图中的字图g，g和查询图P是同构的。文献中无标签的子图匹配问题称作子图迭代，也叫做子图同构搜索，子图列举。）

In the literature, subgraph enumeration is also referred to as ：

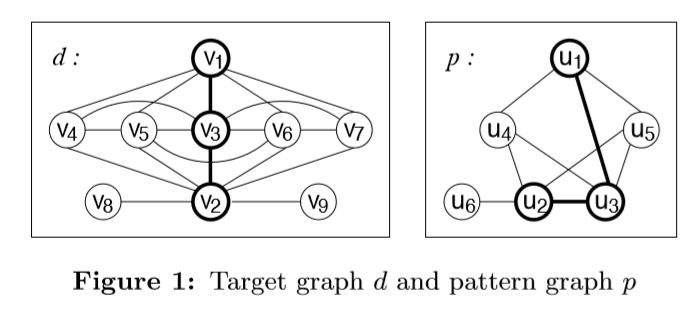
* + subgraph isomorphism search

X. Ren and J. Wang. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs.(VLDB15’)

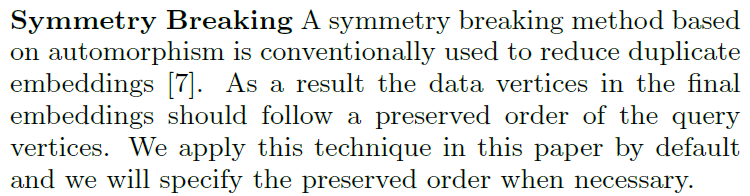
* + subgraph listing

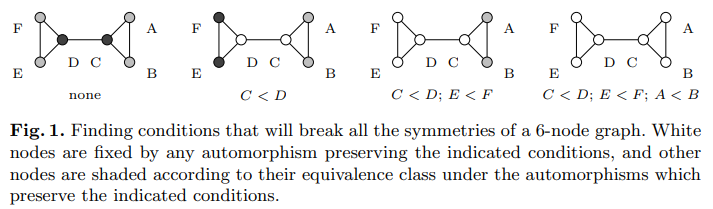
Y. Shao, B. Cui, L. Chen, L. Ma, J. Yao, and N. Xu.Parallel subgraph listing in a large-scale graph. (SIGMOD14’)

(Automorphism) Given a graph g, an automorphism of g is a match from g to itself. We use A(g) to denote the set of automorphisms for a graph g.



（之前介绍的图的同构问题实际上是从一个图到另一图的映射，图的自同构则是一个图本身的映射。比如右边图p，u4和u5位置相同，交换位置就形成了一个自同构。自同构问题是子图匹配问题中必须解决的，假定查询图p的两个匹配分别<v1,v2,v3,v4,v5,v6>，<v1,v2,v3,v5,v4,v6>，由于u4和u5的位置结构相似，属于一种结构。因此，这2个匹配实际增加了我们的输出以及计算代价。因此在无标签的子图迭代问题中默认采用了symeter Breaking技术。目的是解决图中的对称特性。比如下面这个图，相同位置……）





# challenge：

1. unlabeled, undirected graph, the core operation of subgraph enumeration, i.e. subgraph isomorphism, is NP-complete which brings high computational complexity to the problem
2. memory limit，the sizes of the partial and ﬁnal matching results can be much larger than the data graph itself

第二部分：每个人介绍按照7分钟准备。

陈波冯：

# challenge and setting：(1-2分钟)

chagllenge

1. np-hard
2. data graph is large

setting:

* **Single Machine Algorithms:**

Ullmann’s algorithm[J.ACM’76]

VF2[IEEE Trans’04]

QuickSI[VLDB’08]

TurboIso[SIGMOD’13]

BoostIso[VLDB’15]

The increasing size of modern graph makes it hard to load the whole graph into memory.

Distributed Settings:

BFS:

1. decompose the pattern graph recursively into a series of join units.

Star[VLDB’12]

TwinTwig[VLDB’15]

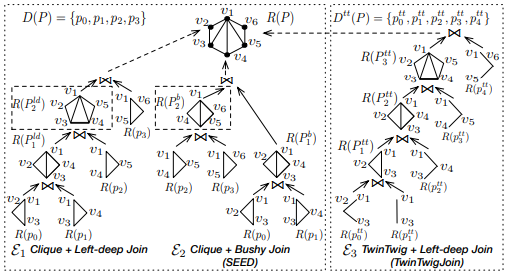
SEED[VLDB’16]

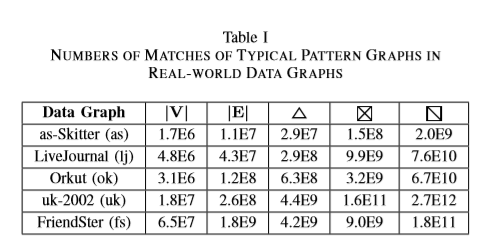
(首先将查询图分解成一系列的子图，star是一个顶点和它所有的边。twinTwig是 2条边，第三个可以三角形)

1. They enumerate the matching results of the join units and assemble them via one or more rounds of join to get the matching results for the whole pattern graph

(left-dep join [12], Bushy join [5], Hash-assembly [6] and Generic join [13]

Problem：shuffling partial matching results is inevitable in the join based framework, causing high communication costs



（分别匹配了1的join单元，第二步就是迭代匹配的结果进行连接，得到最终结果）

索引构建浪费空间。

DFS:

QFrag [10]: M. Seraﬁni, G. D. F. Morales, and G. Siganos, “Qfrag: distributed graph search via subgraph isomorphism,” in Proceedings of the 2017 Symposium on Cloud Computing, 2017, pp. 214–228. 图复制到所有机器

F. N. Afrati, D. Fotakis, and J. D. Ullman, “Enumerating subgraph instances using map-reduce,” in 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 62–73.

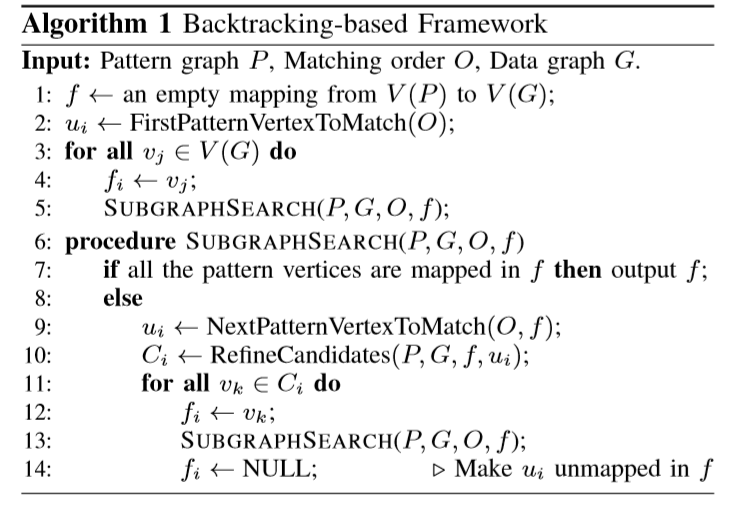
不需要shuffle中间结果，但是需要复制大量的边到所有机器。

动机：如何避免shuffle中间结果？减少甚至避免使用索引？

# 2. 问题定义与模型框架：on-demand shuffle，模型框架。（1分钟）

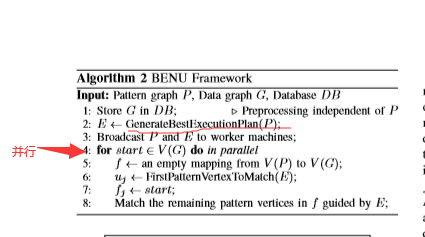
定义：之前相同

dfs框架：



（这种dfs很有问题。考虑：查询图中包含一个三角形，但是数据图中没有这个三角形结构，那么一种更有效的方式是去先看有没有三角形再去做子图匹配。

on-demand shufﬂe: 主要思想是将数据图存储在分布式数据库中，在RefineCandidates需要这条边的时候，去请求。）

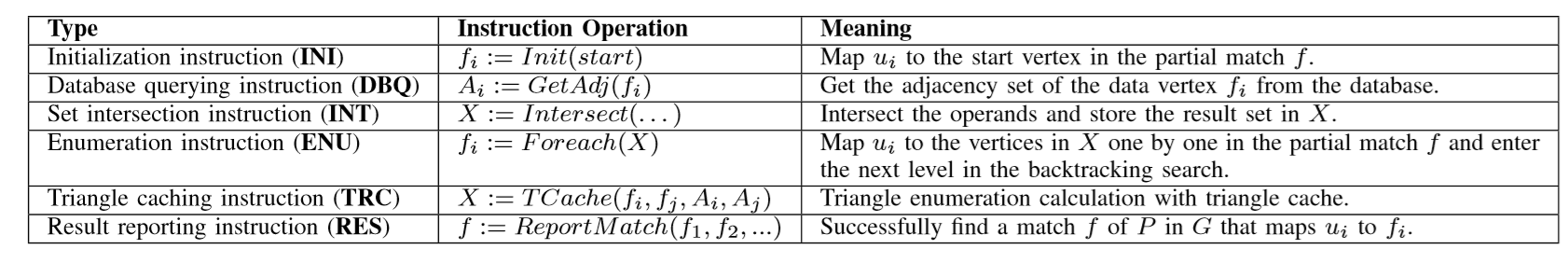


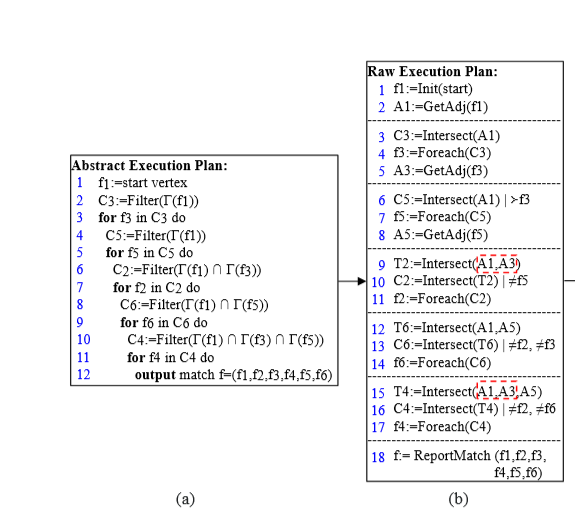
（基于以上想法定义了一个BENU框架。）

# 3.贡献点：（4分钟）

1. Raw Execution Plan Generation：

Given a matching order O: uk1,uk2,...,ukn, the raw execution plan consists of a series of execution instructions.





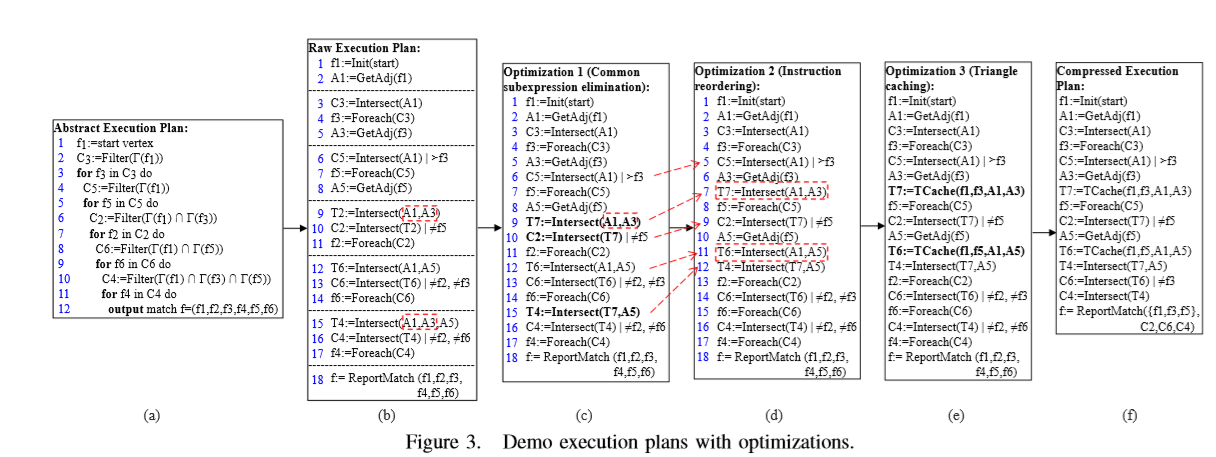
介绍每个指令的作用。

2.

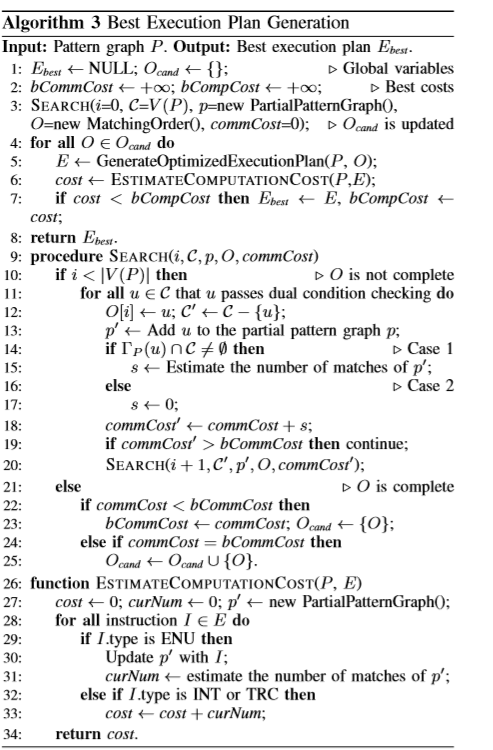
Optimization 1 (Common subexpression elimination).

Optimization 2 (Instruction reordering).

Optimization 3 (Triangle caching).



1. Execution Plan Cost Estimation and Best Execution Plan Generation



4.EFFICIENT IMPLEMENTATION

# 4.实验 （1分钟）

**Environment.** All the experiments were conducted in a cluster with 17 machines (1 master + 16 workers) connected via 1 Gbps Ethernet. Each machine was equipped with 12 cores, 50 Gbytes memory and 2 Tbytes RAID0 HDD storage. All the algorithms were implemented with Hadoop 2.7.2, compiled with JDK 1.8 and run in CentOS 7.0. The map output in Hadoop was compressed with Gzip.

Data Graphs：

Pattern Graphs.

1. Evaluation of Optimization Techniques in BENU

B. Compare with Existing Approaches

# 5.总结和个人思考（1分钟）