Scalable Subgraph Enumeration in MapReduce

Authors Longbin Lai ● Lu Qin ●
Xuemin Lin ● **Lijun Chang** ●

● *CSE, The University of New South Wales, Australia*
● *QCIS, University of Technology, Sydney*

Problem Statement

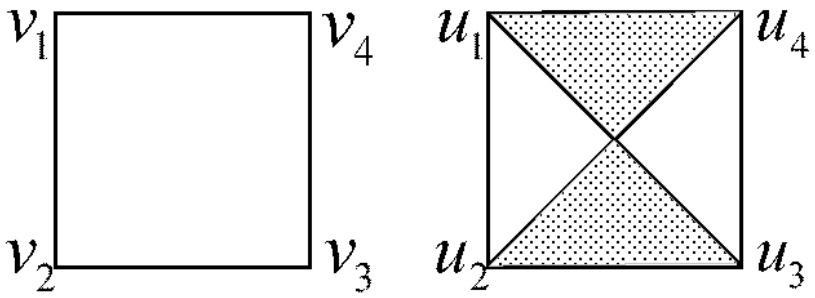
Given a pattern graph P, we aim at finding all matches of P in a large data graph *G* using MapReduce

Both *P* and *G* are **unlabeled**, **simple** graphs

G is arranged as $(u; N(u))$ for each data nodes, and stored in the HDFS

Subgraph Enumeration

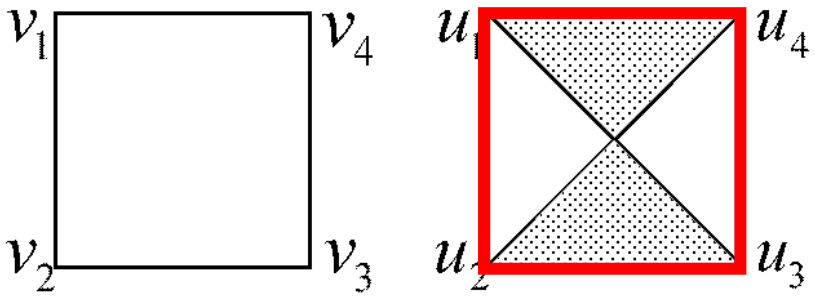
Given a pattern graph P, we aim at finding all matches of P in a large data graph G using MapReduce



P *G*

Subgraph Enumeration

Given a pattern graph P, we aim at finding all matches of P in a large data graph G using MapReduce



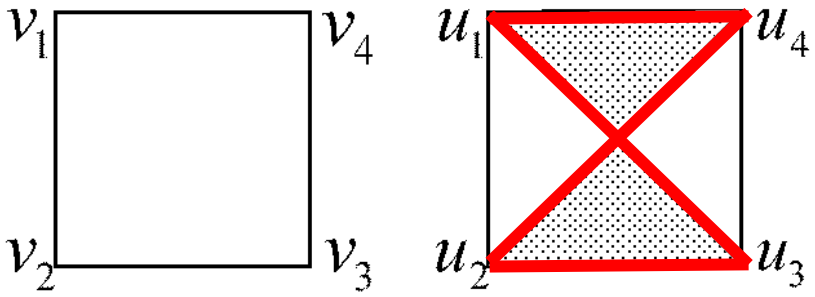
P *G*

We can find three matches

(u_1, u_2, u_3, u_4), the peripheral square

Subgraph Enumeration

Given a pattern graph P, we aim at finding all matches of P in a large data graph G using MapReduce



P G

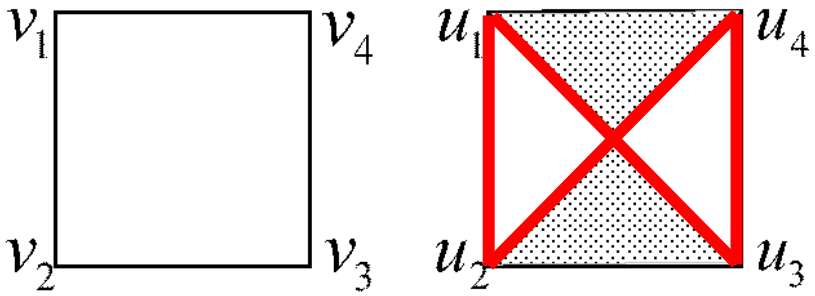
We can find three matches

(u_1, u_2, u_3, u_4), the peripheral square

(u_1, u_3, u_2, u_4), the shadowed part

Subgraph Enumeration

Given a pattern graph P, we aim at finding all matches of P in a large data graph G using MapReduce



P G

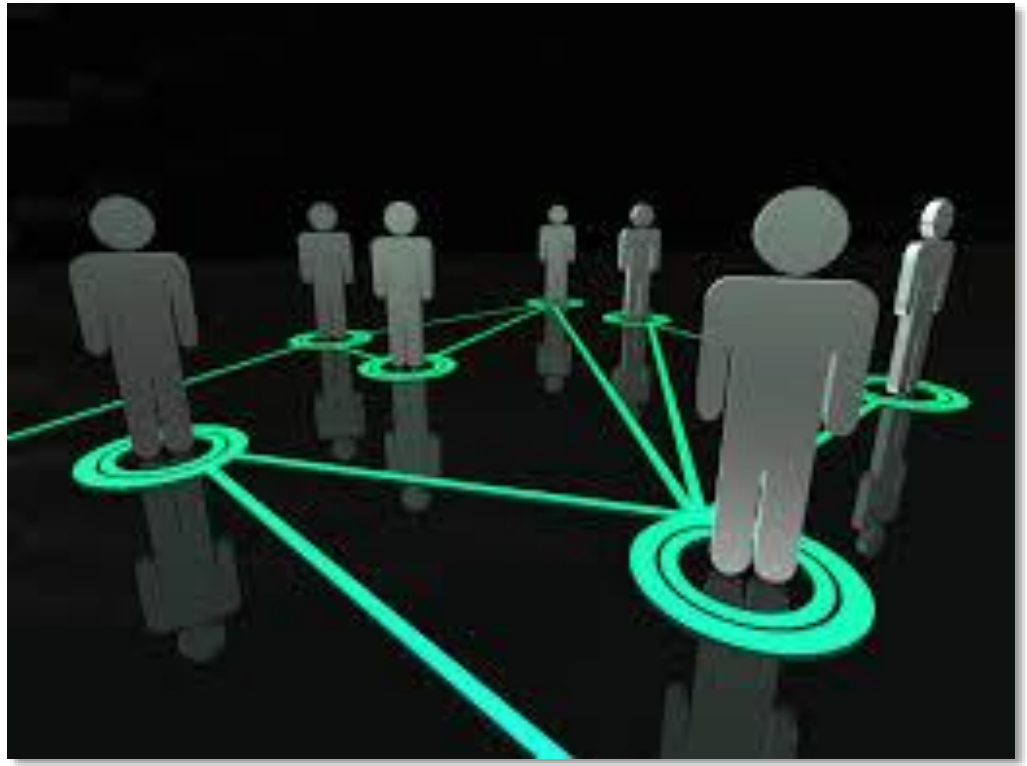
We can find three matches

(u_1, u_2, u_3, u_4), the peripheral square

(u_1, u_3, u_2, u_4), the shadowed part

(u_1, u_2, u_4, u_3), the white part

Why Subgraph Enumeration



Social Networks

Community Detection

Nodes in the same communities tend to form specific subgraphs (clique, e.g.)

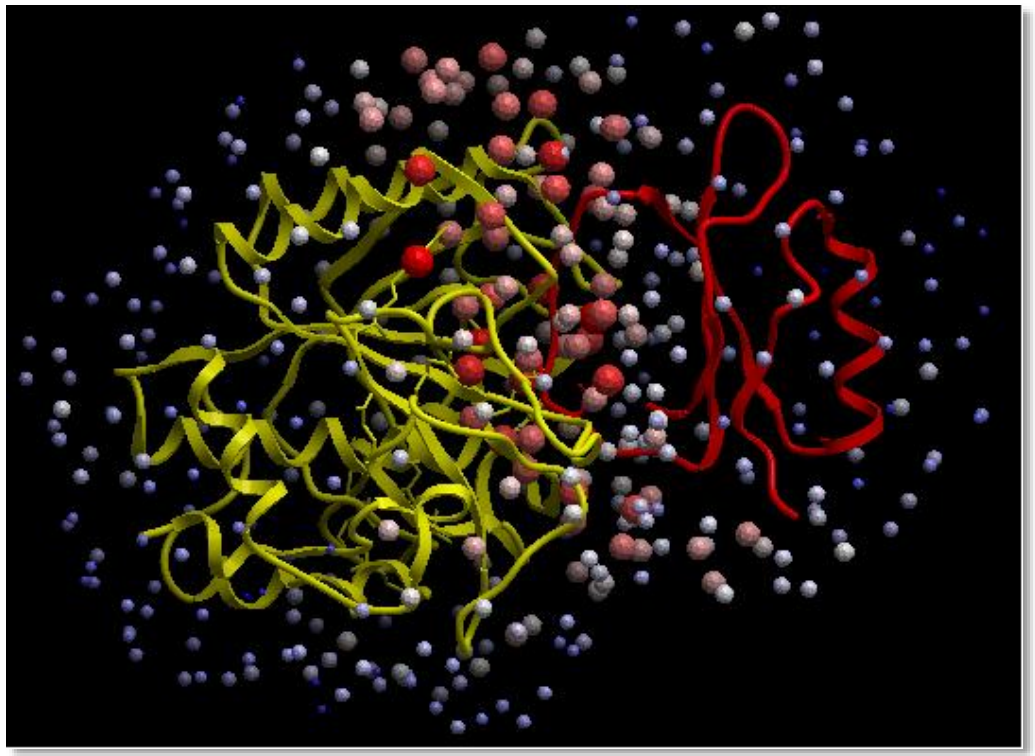
Network Similarity Study

The frequencies of specific pattern graphs can be good features to study network similarity

Cluster Coefficient Calculation

Triangles / (# neighbors X (#neighbors - 1))

Why Subgraph Enumeration



BioInformatics

Motif Discovery

Motifs are some very frequent patterns

Complex Network Design

Specific pattern graphs can facilitate complex network design

Abnormity Detection

Abnormity represents as unexpected statistical information of specific pattern graphs

Most Related Work

- **Edge Join [T. Plantenga, JPDC 2012]**
  - Join edges one by one to get the final results
  - Finish in m rounds, not scalable to handle complicated pattern graph
  - Too many intermediate results

- **Multiway join [F.Afrati, D.Fotakis, J.D.Ullman, ICDE 2013]**
  - One-round computation by duplicating edges
  - Do not scale well due to memory issues

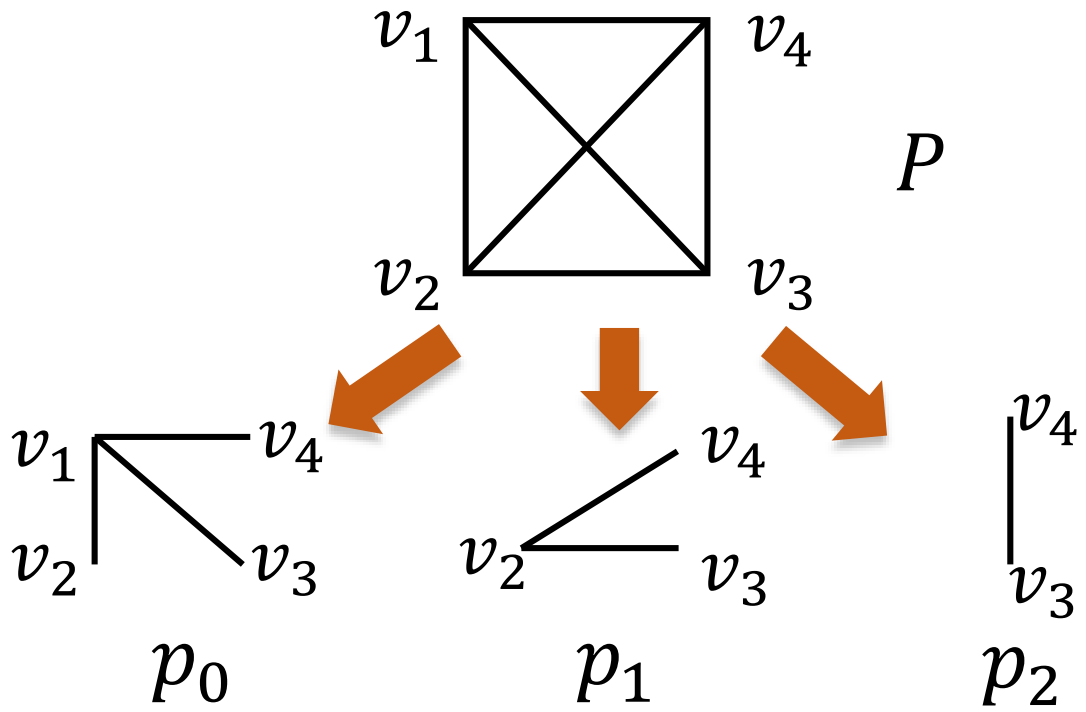
c 1: Een Goed begin is het halve werk

- Problem Statement
- **Methodology**
- Evaluation
- Conclusions

Progress: 25%

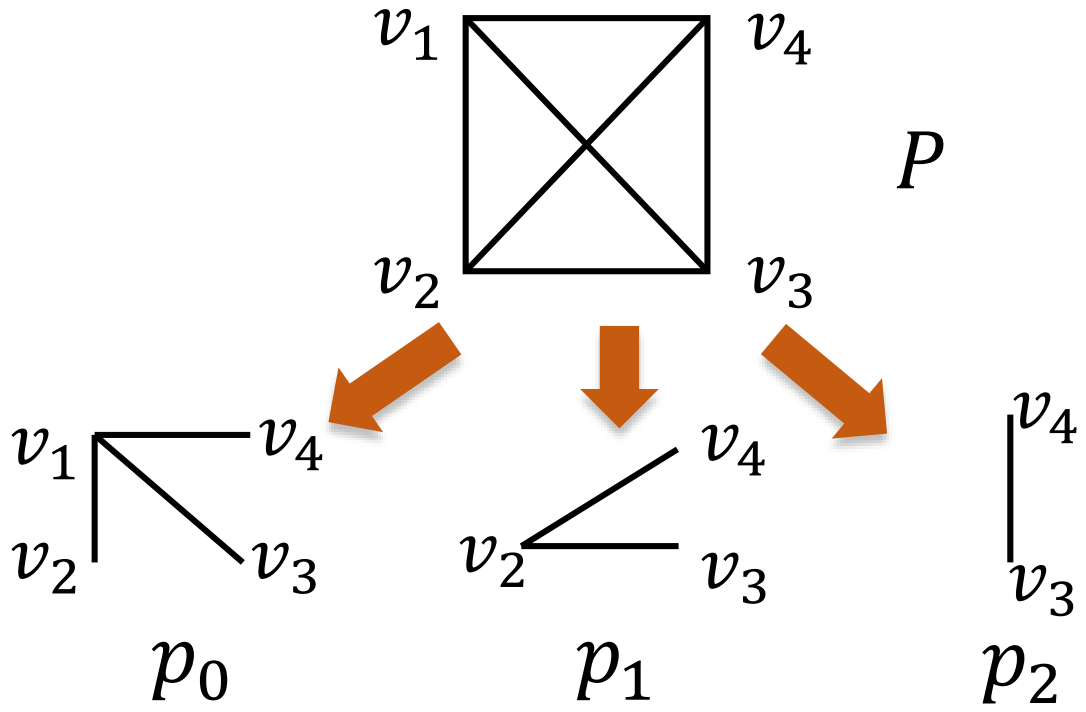
Graph Decomposition

Pattern Graph is decomposed into a sequence of **stars**



Graph Decomposition

Pattern Graph is decomposed into a sequence of **stars**

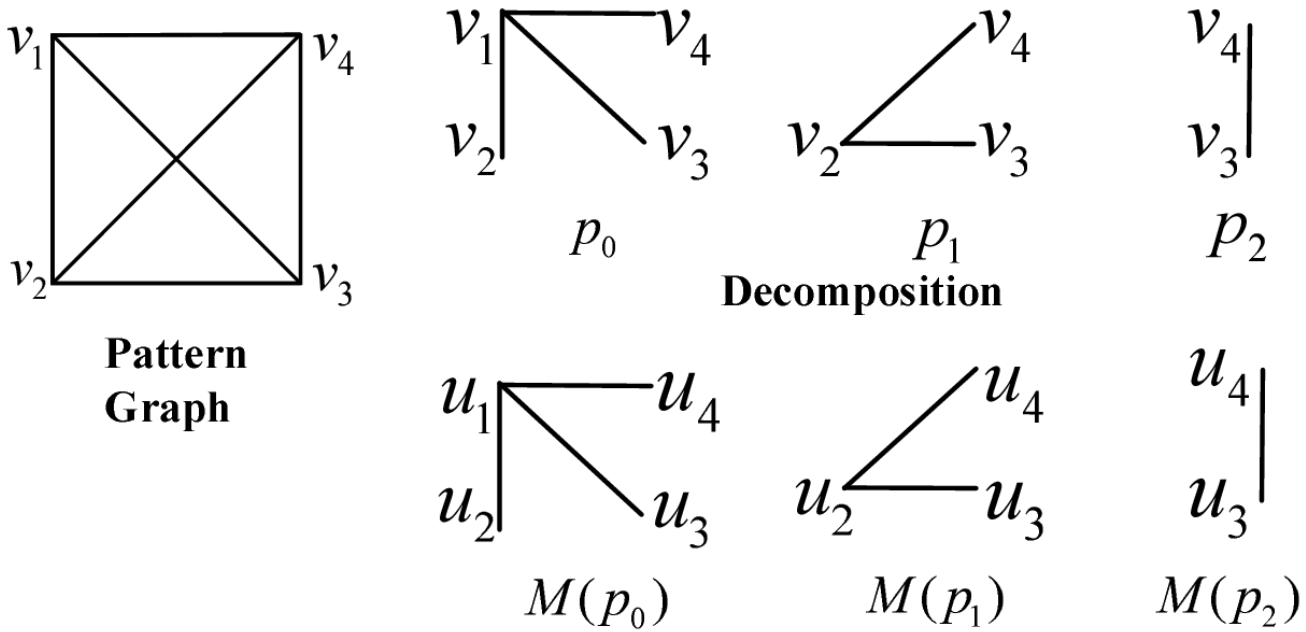


Why Star?

When we have $(u, N(u))$ as input, we can search the matches of star rooted on u by enumerating the node combinations in $N(u)$.

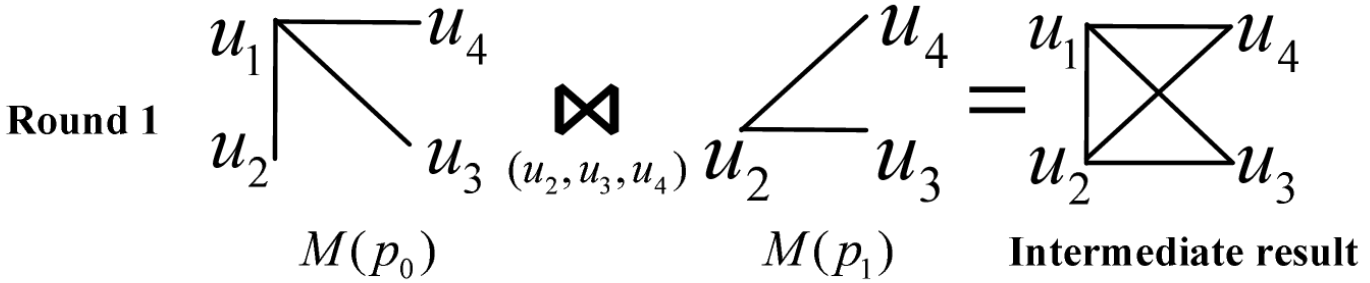
Left-deep Join Framework

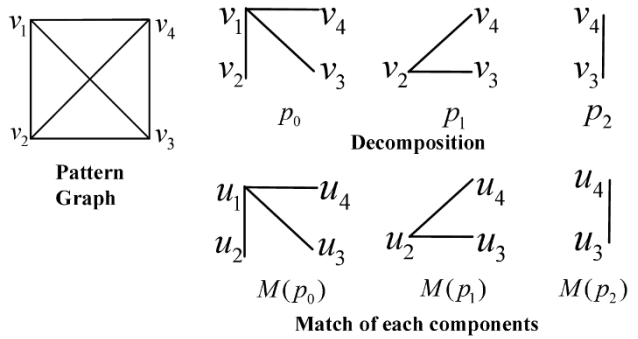
After decomposition, we will process (t-1)-round left-deep join operations using MapReduce

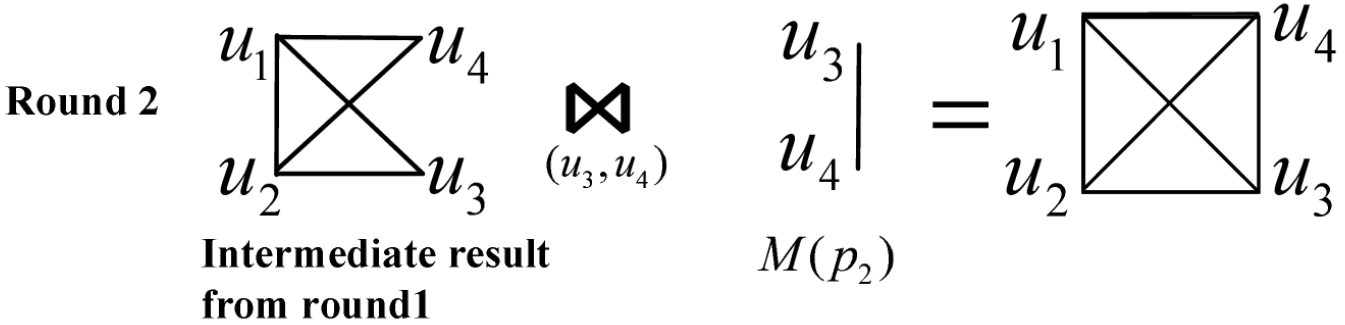


Left-deep Join Framework

After decomposition, we will process (t-1)-round left-deep join operations using MapReduce

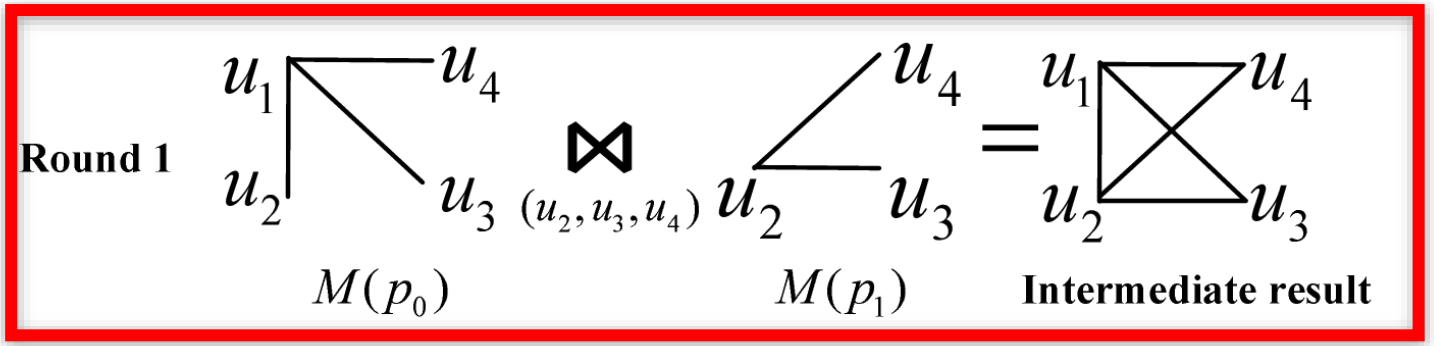


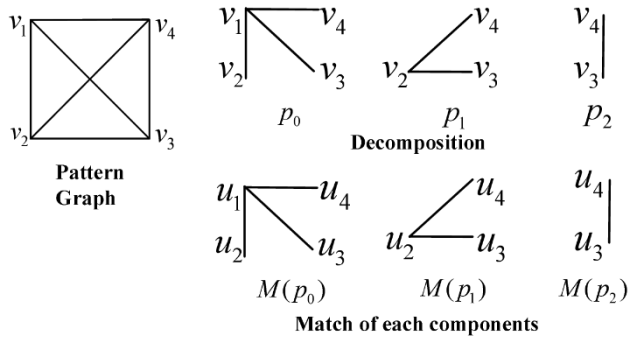


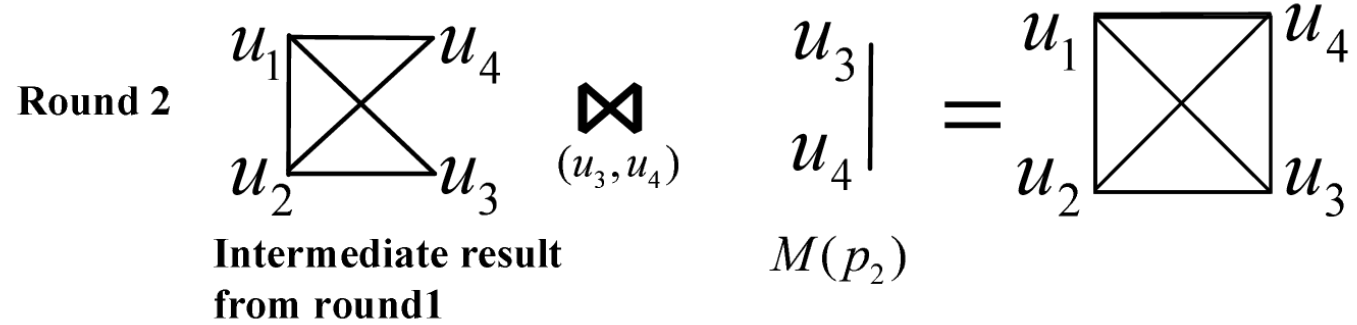


Left-deep Join Framework

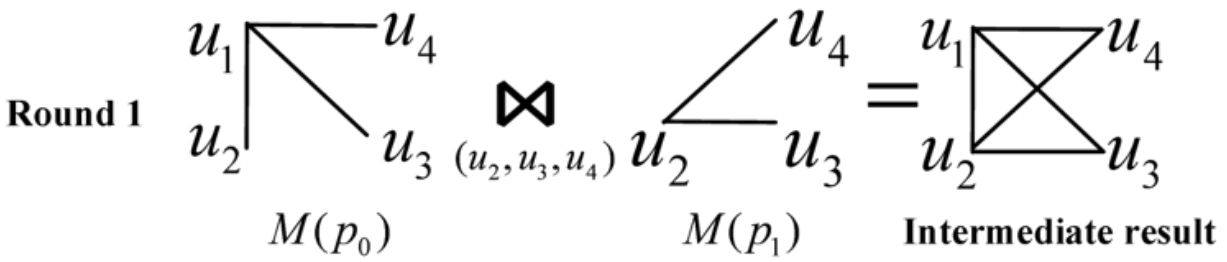
After decomposition, we will process (t-1)-round left-deep join operations using MapReduce

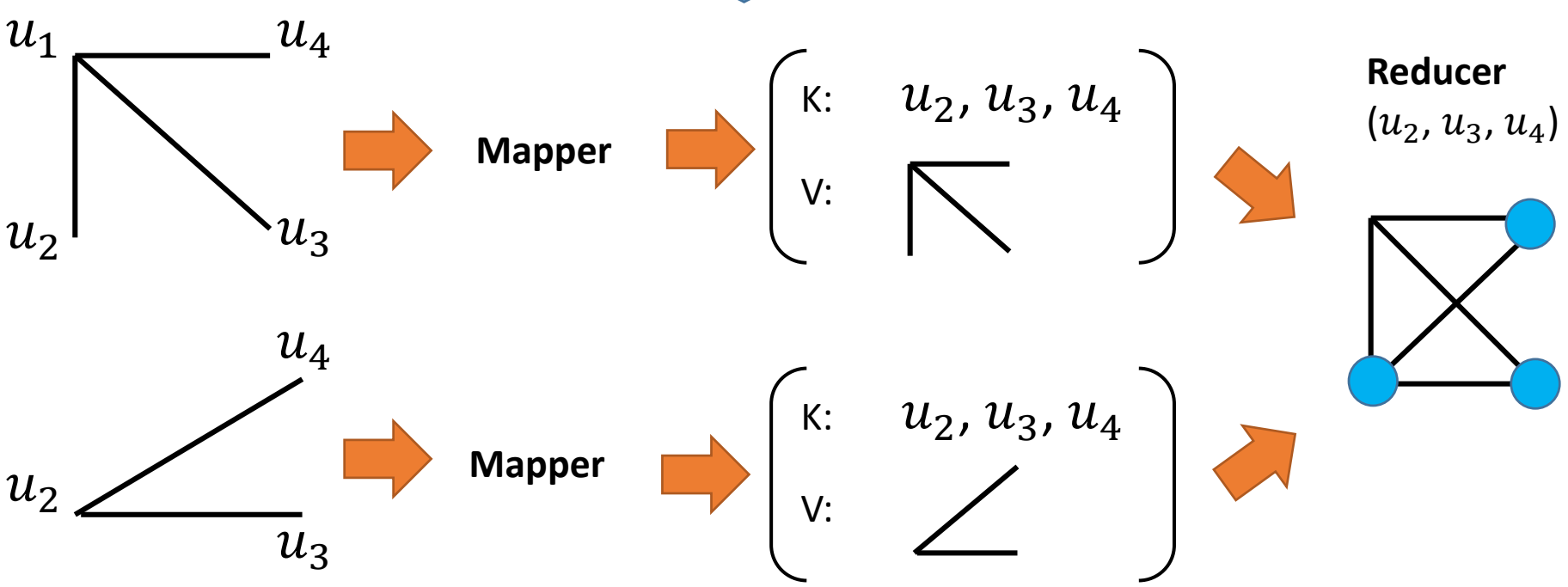






Left-deep Join Framework





Matching a star can still produce large number of intermediate results

TwinTwig

Matching a star can still produce large number of intermediate results

Common for a node to contain many neighbors in a large social networks or web graphs

A node with 100,000 neighbors => 10^{15} matches of a 3-star

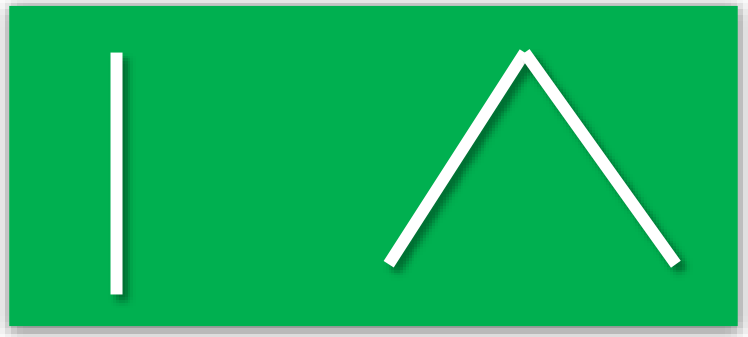
TwinTwig

Matching a star can still produce large number of intermediate results

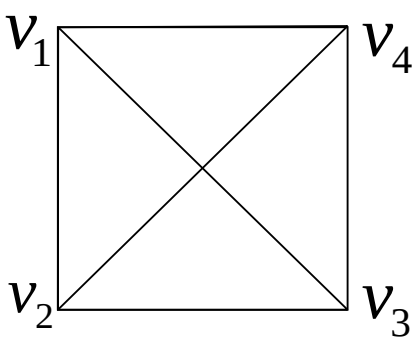
Common for a node to contain many neighbors in a large social networks or web graphs

TwinTwig

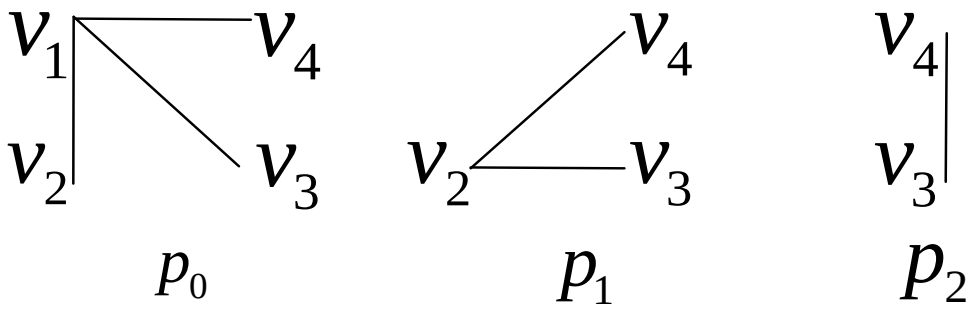
A star of **at most** two edges



TwinTwig Decomposition

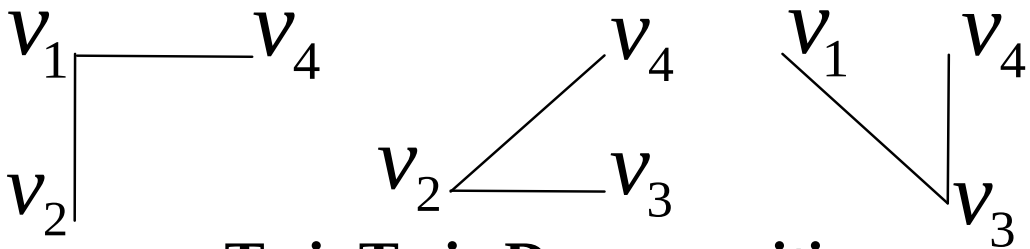


P



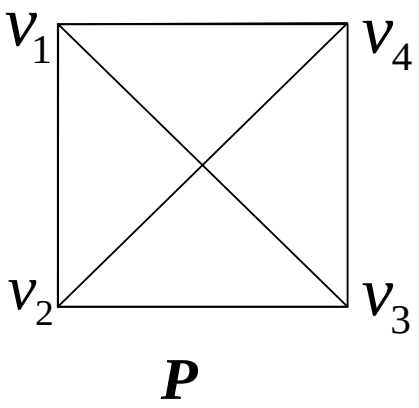
p_0 p_1 p_2

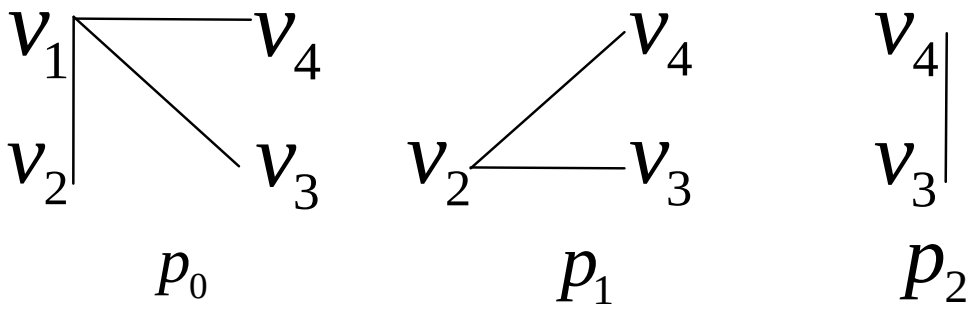
Star Decomposition



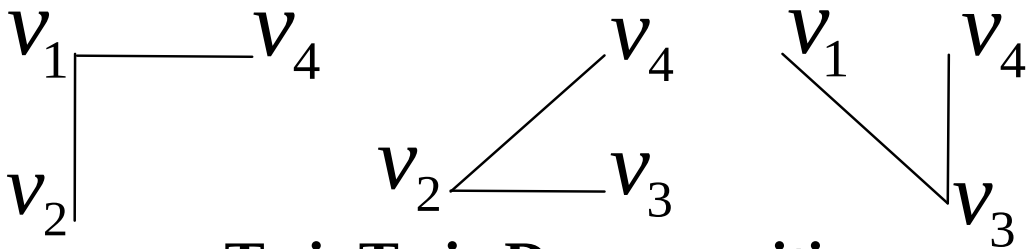
TwinTwig Decomposition

TwinTwig Decomposition





Star Decomposition

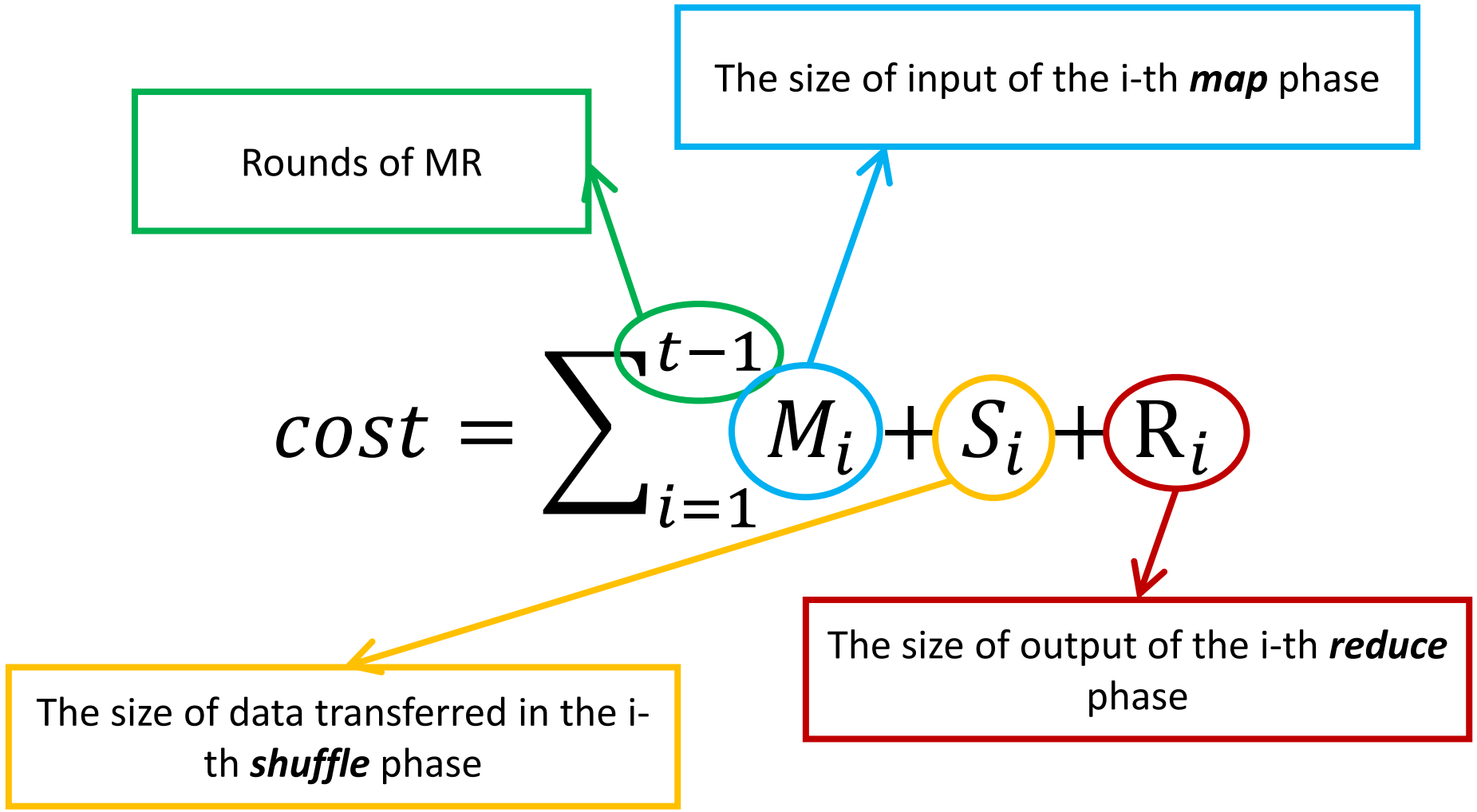


TwinTwig Decomposition

Star Join: The join process w.r.t. star decomposition

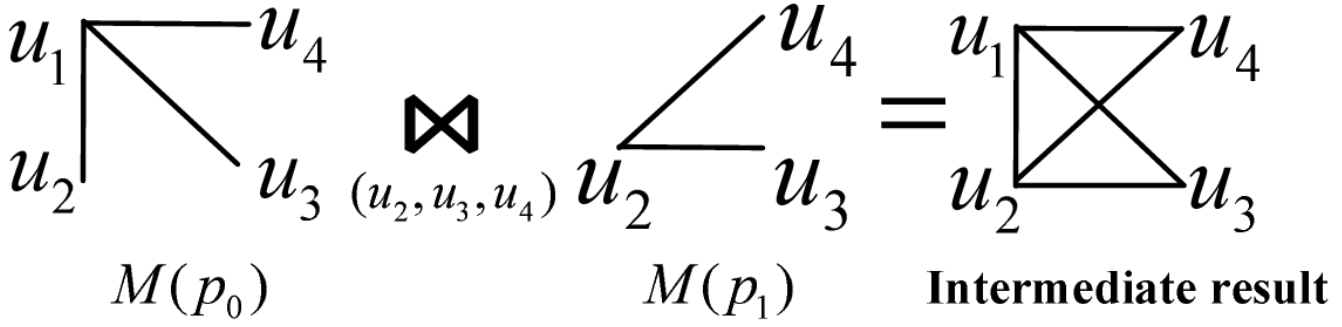
TwinTwig Join: The join process w.r.t. TwinTwig decomposition

Cost Model


Rounds of MR
The size of input of the i-th *map* phase
$cost = \sum_{i=1}^{t-1} M_i + S_i + \mathrm{R}_i$
The size of data transferred in the i-th *shuffle* phase
The size of output of the i-th *reduce* phase

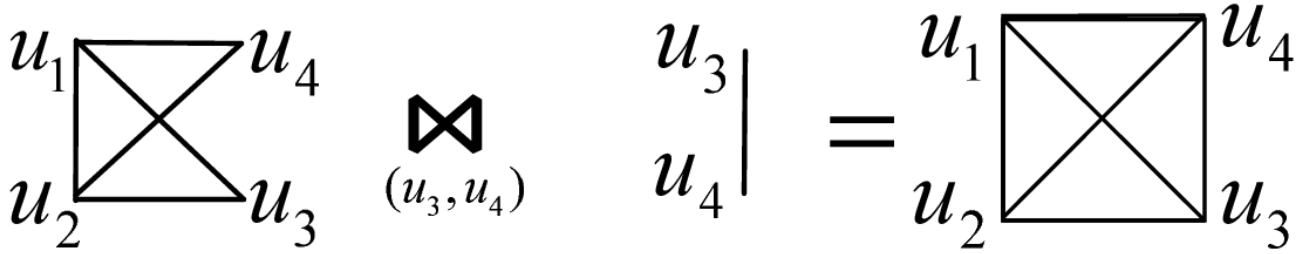
Cost Model

Round 1



$M(p_0)$ $M(p_1)$ **Intermediate result**

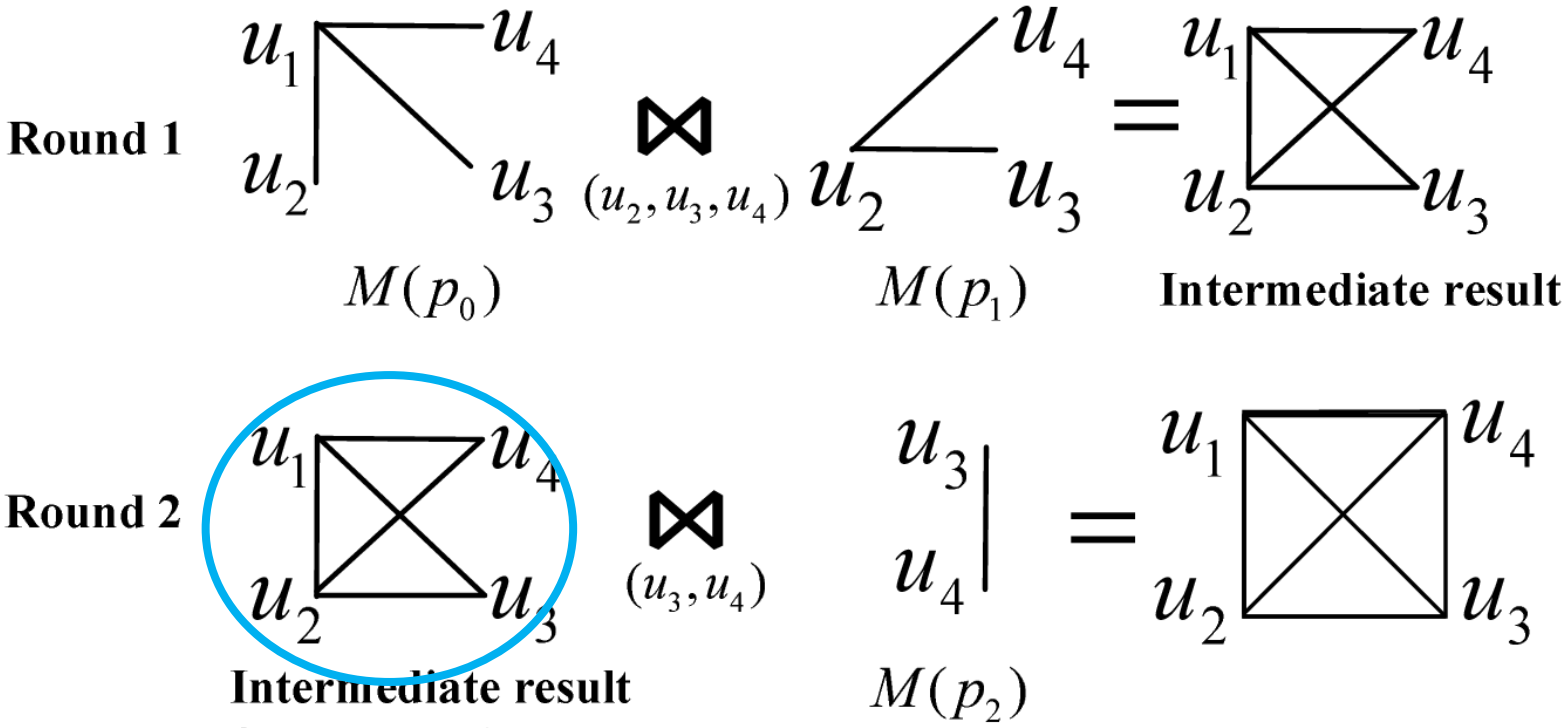
Round 2



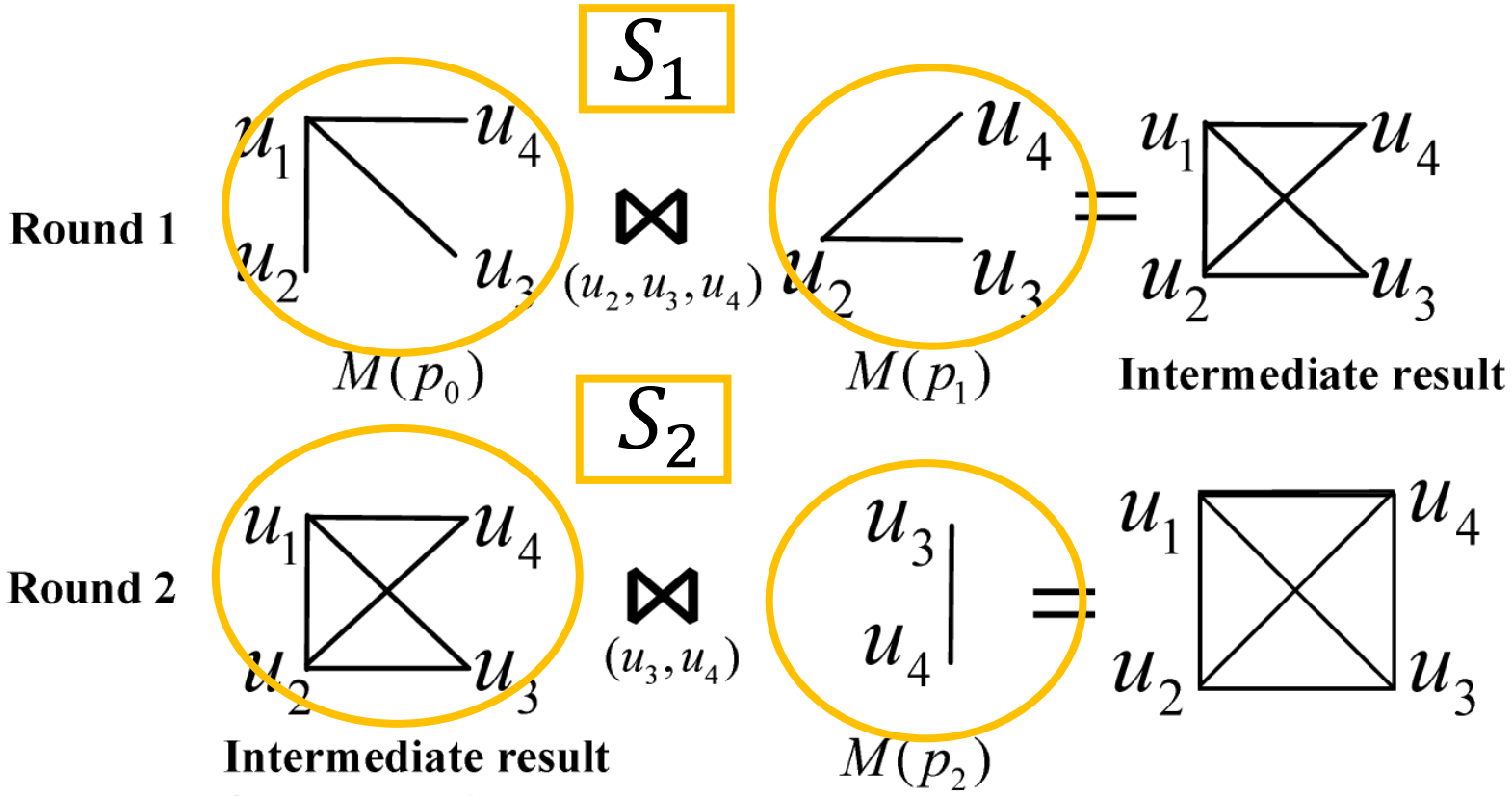
Intermediate result from round1

$M(p_2)$

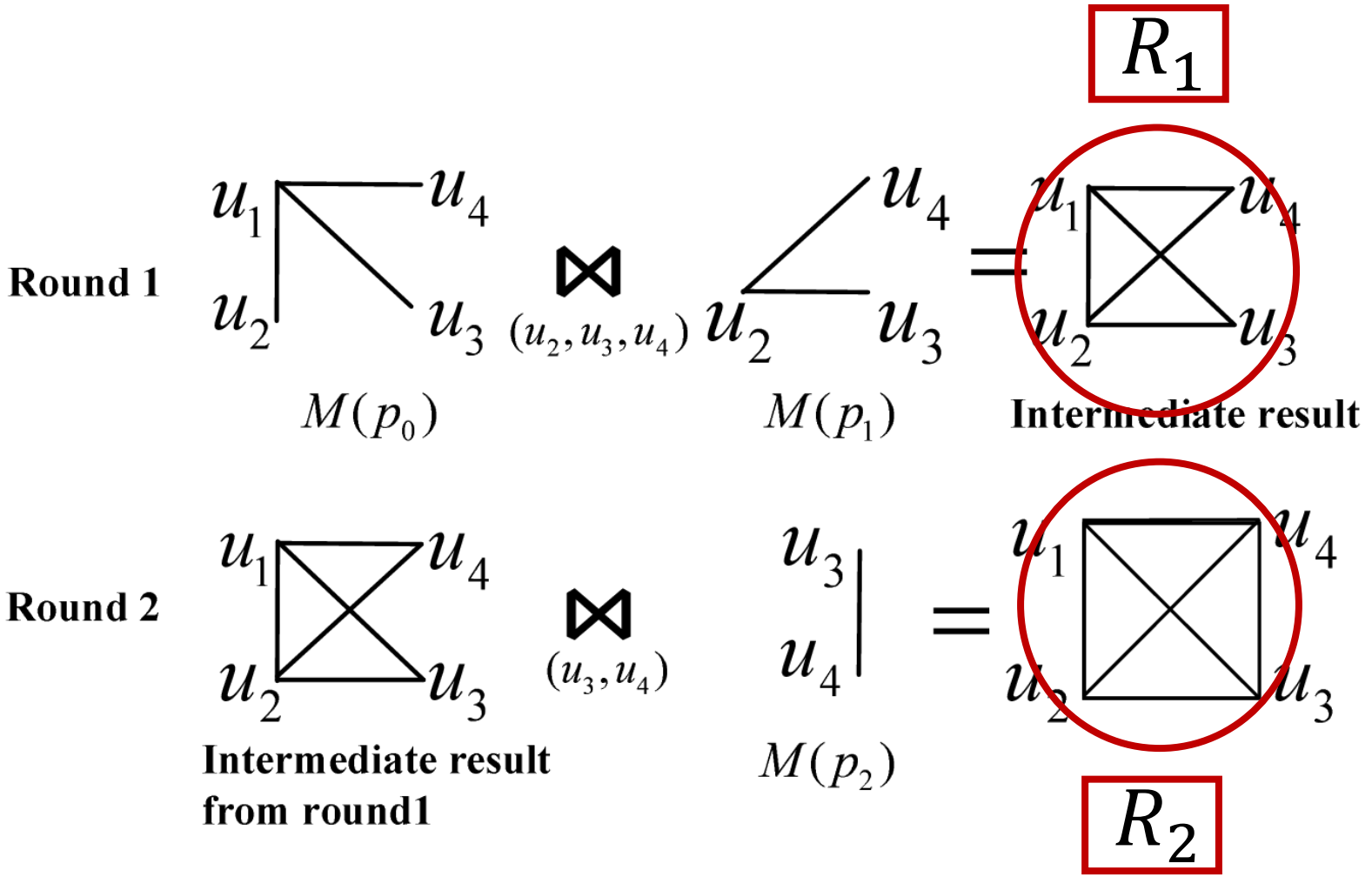
Cost Model


Round 1
u_1
u_4
u_2
u_3
$M(p_0)$
$\bowtie_{(u_2,u_3,u_4)}$
u_4
u_2
u_3
$M(p_1)$
=
u_1
u_4
u_2
u_3
Intermediate result
Round 2
u_1
u_4
u_2
u_3
Intermediate result
from round1
$\bowtie_{(u_3,u_4)}$
u_3
u_4
$M(p_2)$
=
u_1
u_4
u_2
u_3

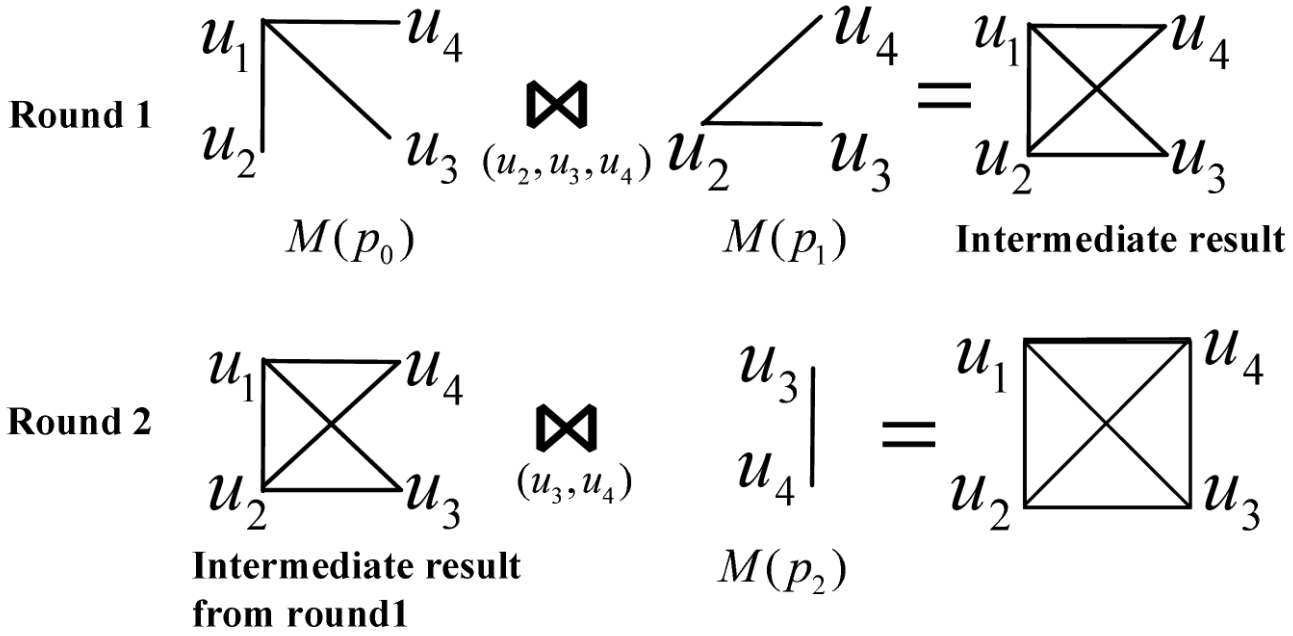
Cost Model


S_1
Round 1
u_1
u_4
u_2
u_3
$M(p_0)$
$\bowtie_{(u_2,u_3,u_4)}$
u_4
u_2
u_3
$M(p_1)$
=
u_1
u_4
u_2
u_3
Intermediate result
S_2
Round 2
u_1
u_4
u_2
u_3
Intermediate result
from round1
$\bowtie_{(u_3,u_4)}$
u_3
u_4
$M(p_2)$
=
u_1
u_4
u_2
u_3

Cost Model


R_1
Round 1
u_1
u_4
u_2
u_3
(u_2, u_3, u_4)
$M(p_0)$
$M(p_1)$
Intermediate result
Round 2
u_3
u_4
(u_3, u_4)
Intermediate result
from round1
$M(p_2)$
R_2

Cost Model



In order to estimate the cost, we need to compute the number of matches of any pattern graphs in the data graph before running the algorithms

Random Graph Model

Expected number of matches of *P in G*

$$|R(P)| = \frac{(2M)^m}{N^{2m-n}}$$

G: Random Data graph

P: Pattern graph

M: Number of edges in *G*

N: Number of nodes in *G*

m: Number of edges in *P*

n: Number of nodes in *P*

Instance Optimality

Given a *star join*, *star*, there always exists a *TwinTwig join*, *tt*, s.t.

$$\Theta(cost(tt)) \leq \Theta(cost(star))$$

Instance Optimality

We also show that the instance optimality holds in a power-law random graph model

Experimental Settings

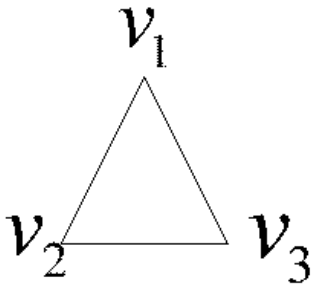
Algorithms	Desc.
Edge	Edge Join
Mul	Multiway Join
Star	Star Join
TT	TwinTwig join

Algorithms

Dataset G	|V(G)|	|E(G)|
sk	1.6M	11.1M
yt	3.2M	12.2M
lj	4.8M	42.9M
orkut	3.1M	261.8M
uk	18.5M	261.8M
fs	65.6M	1806.1M

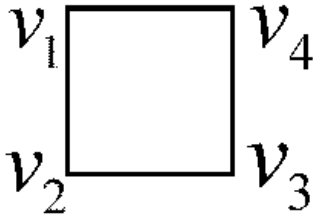
Datasets

Experimental Settings



$v_1 < v_2 < v_3$

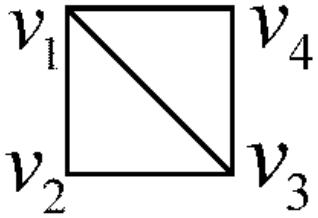
q_1



$v_1 < v_2, v_1 < v_3$

$v_1 < v_4, v_2 < v_4$

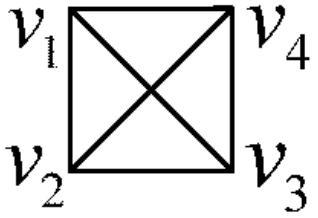
q_2



$v_1 < v_3$

$v_2 < v_4$

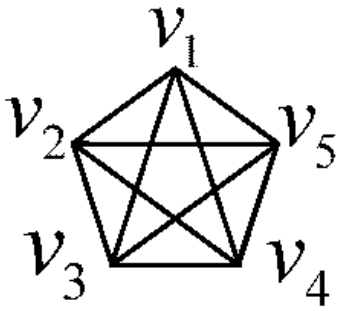
q_3



$v_1 < v_2 < v_3$

$v_3 < v_4$

q_4



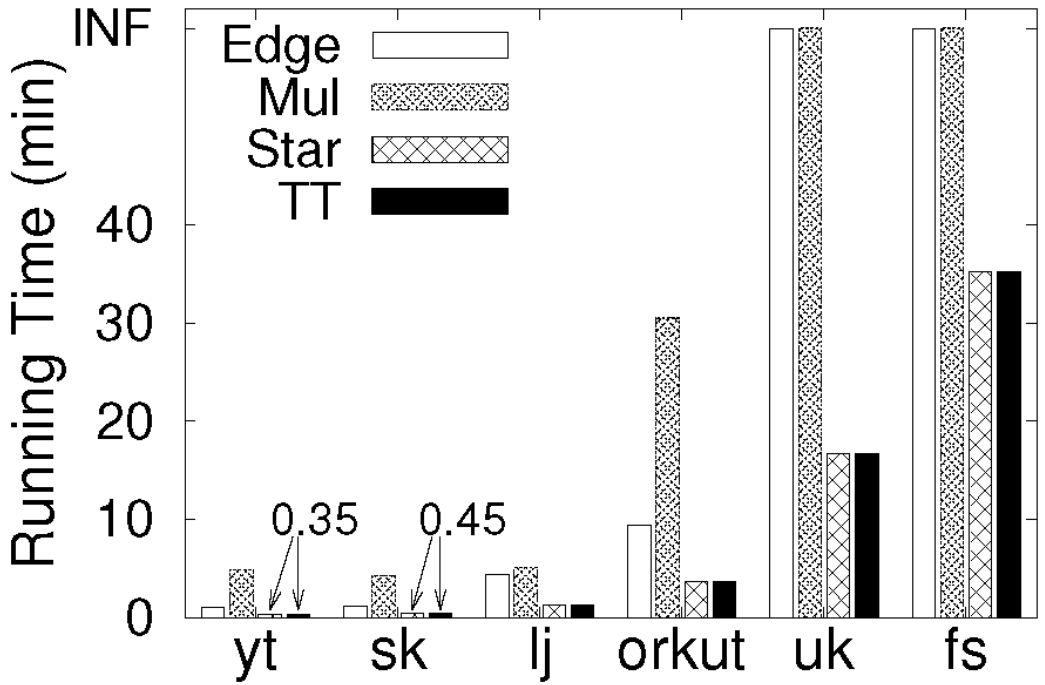
$v_1 < v_2 < v_3$

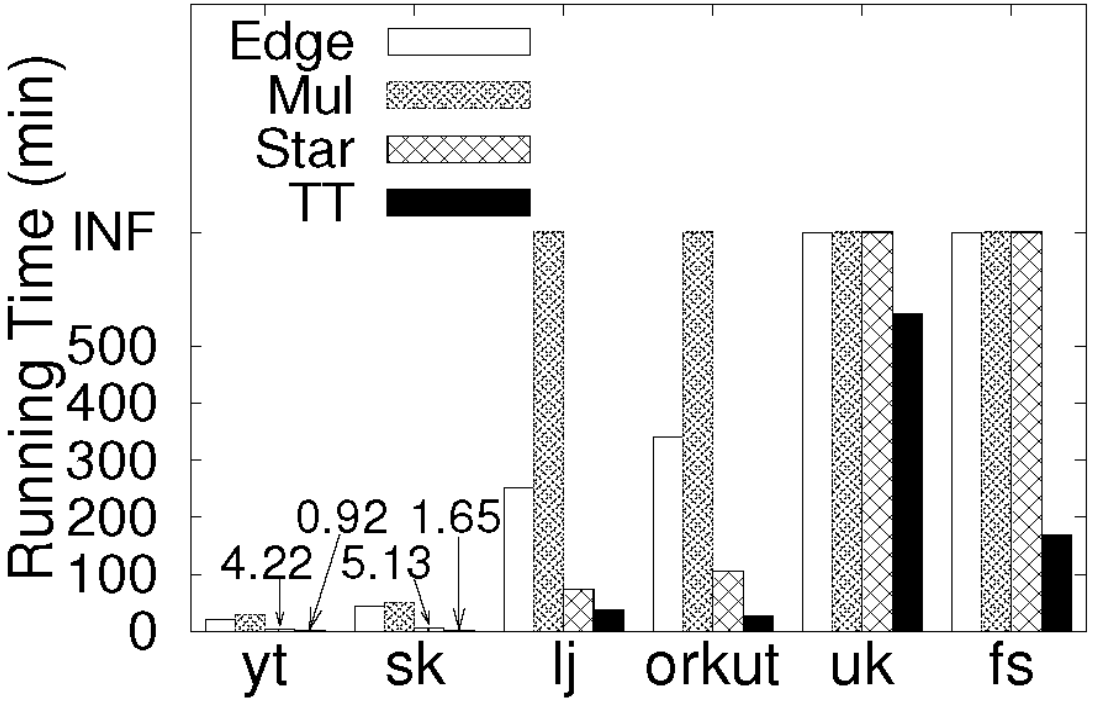
$v_3 < v_4 < v_5$

q_5

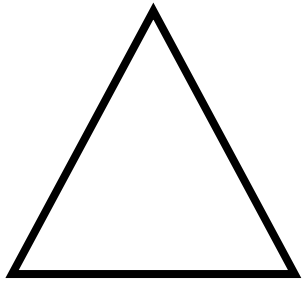
Queries

Vary Datasets

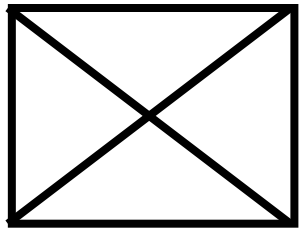




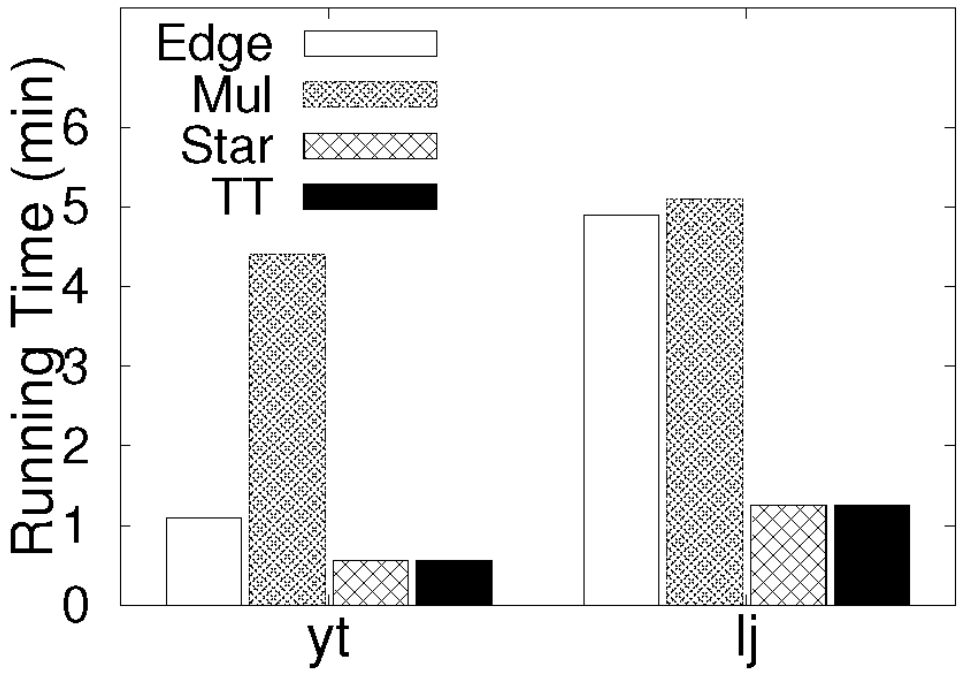
q1

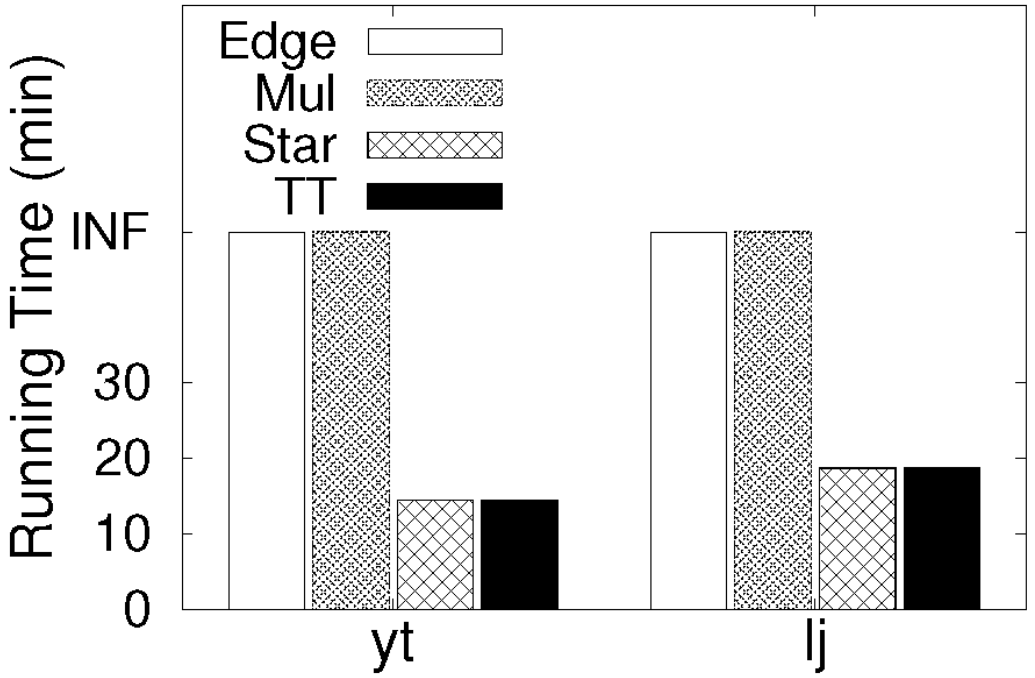


q4

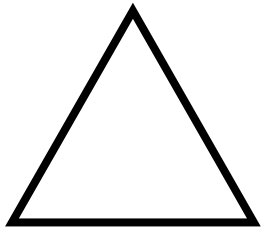


Vary Queries



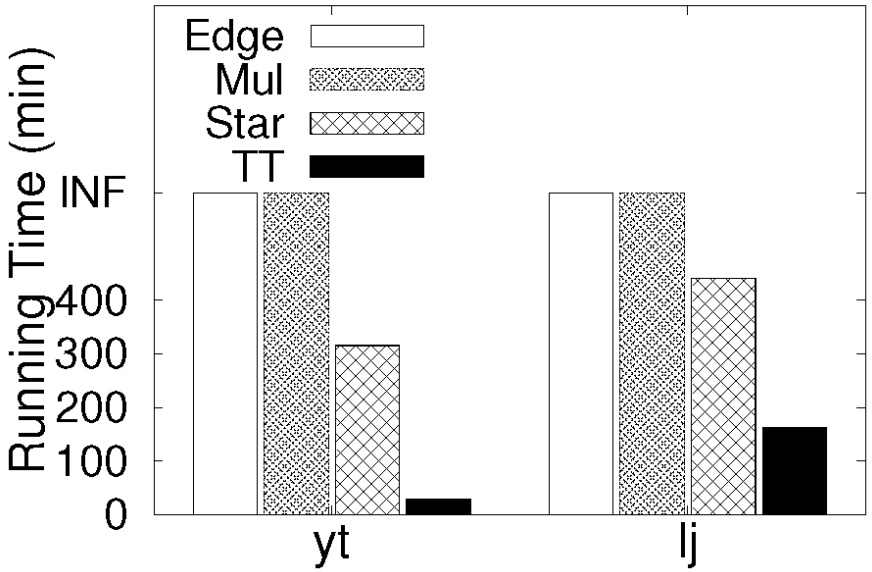


q1

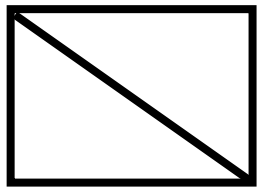


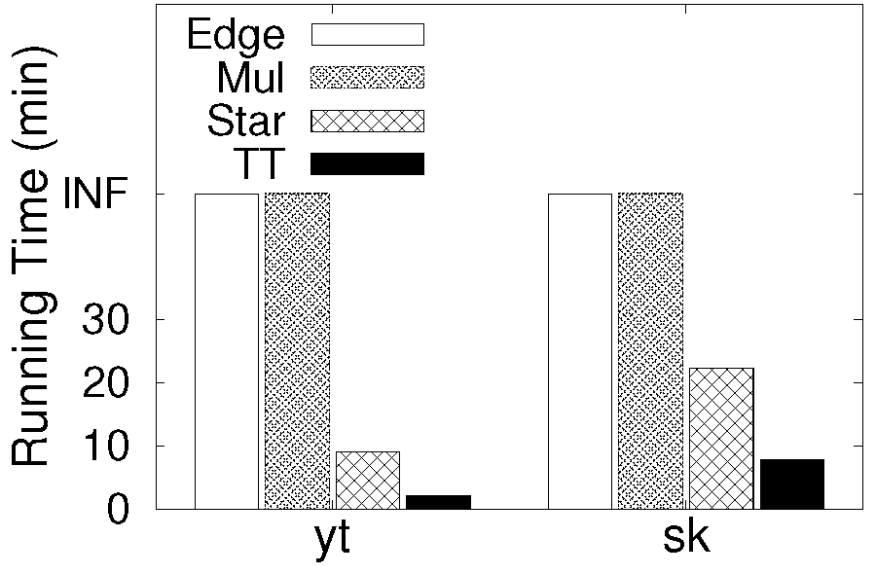
q2

Vary Queries

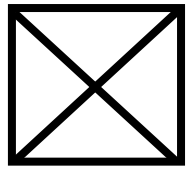


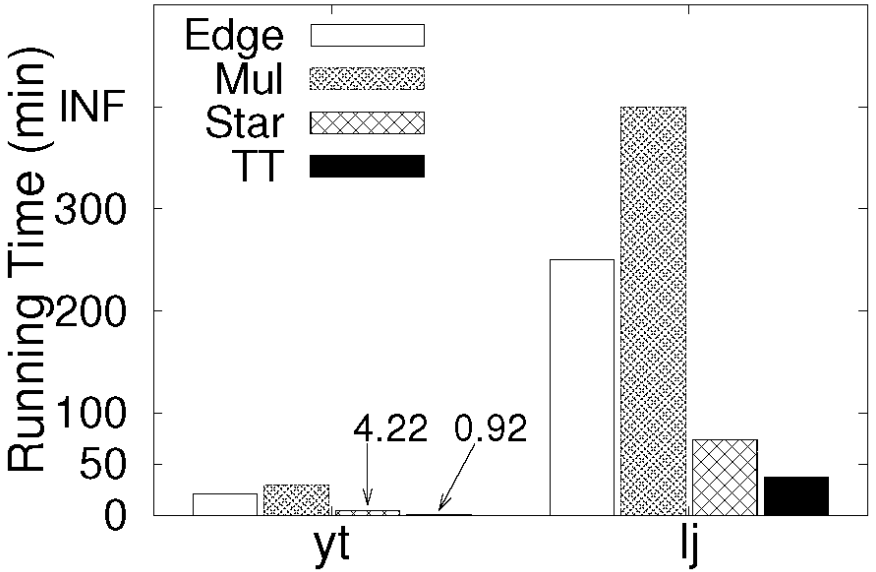
q3



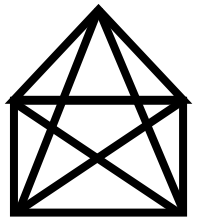


q4





q5



Conclusions

A scalable MapReduce solution for subgraph enumeration

A left-deep-join framework to process star join

Leveraging the (power-law) random graph model, we show the instance optimality of TwinTwig join

Extensive performance studies to confirm the effectiveness of our approach

Thank you & Questions