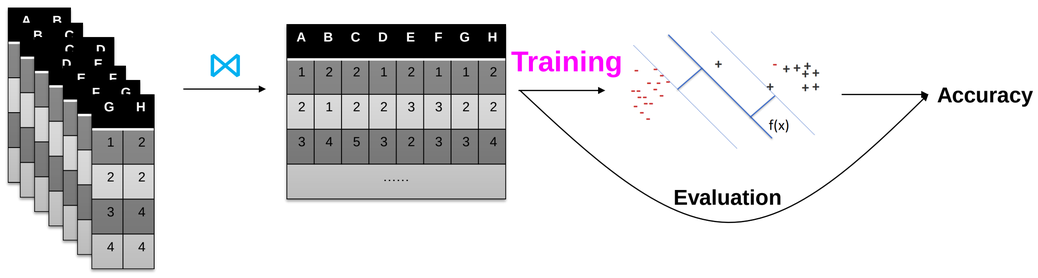
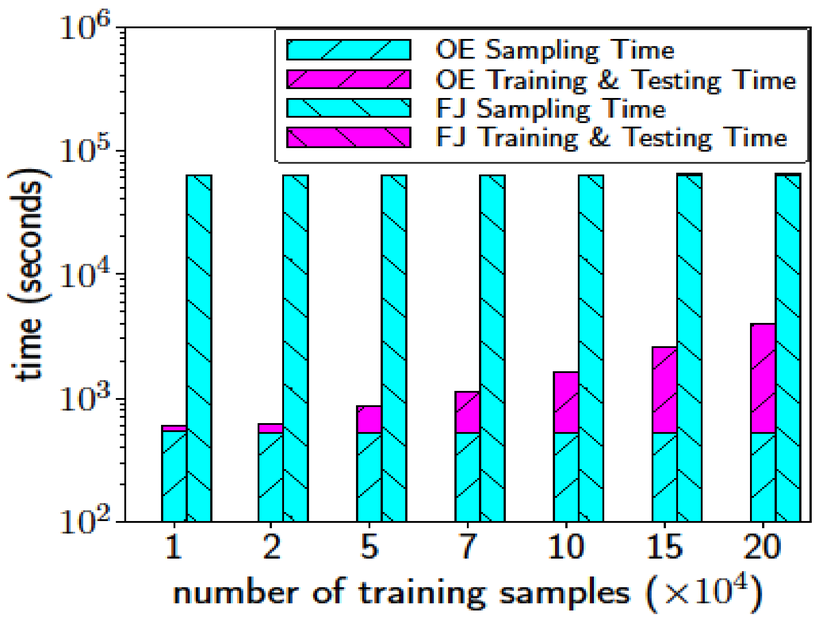
Random Sampling over Joins Revisited

Zhuoyue Zhao[1], Robert Christensen[1], Feifei Li[1], Xiao Hu[2], Ke Yi[2]
{zyzhao, robertc, lifeifei}@cs.utah.edu {xhuam, yike}@cse.ust.hk
University of Utah[1] Hong Kong University of Science and Technology[2]

Motivation

Joins are expensive and random sampling significantly boosts complex analytical tasks (e.g., training SVM over a multi-way join).





OE: a sampling algorithm proposed in this work; FJ: full join and then down-sample

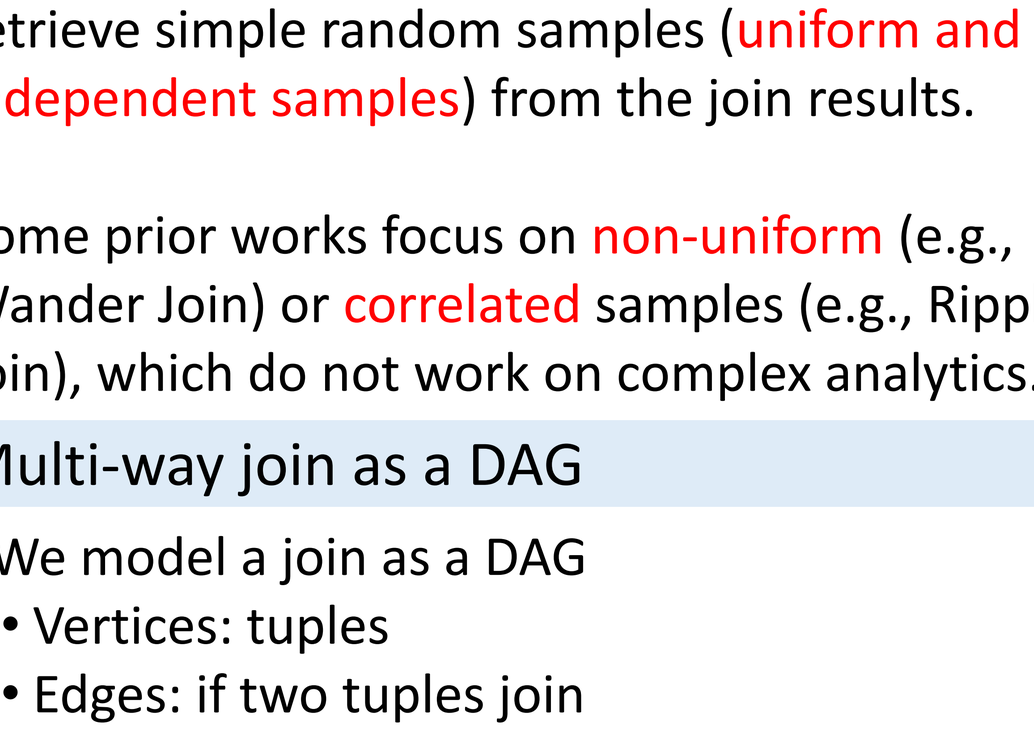
Problem Definition

Given an arbitrary multi-way join, efficiently retrieve simple random samples (uniform and independent samples) from the join results.

Some prior works focus on non-uniform (e.g., Wander Join) or correlated samples (e.g., Ripple Join), which do not work on complex analytics.

Multi-way join as a DAG

- We model a join as a DAG
  - Vertices: tuples
  - Edges: if two tuples join
- Weight of a tuple $w(t)$ = # join results from it
- For simplicity, introduce $R_0 = \{t_0\}$
  - t_0: a "root" tuple that joins any $t_1 \in R_1$

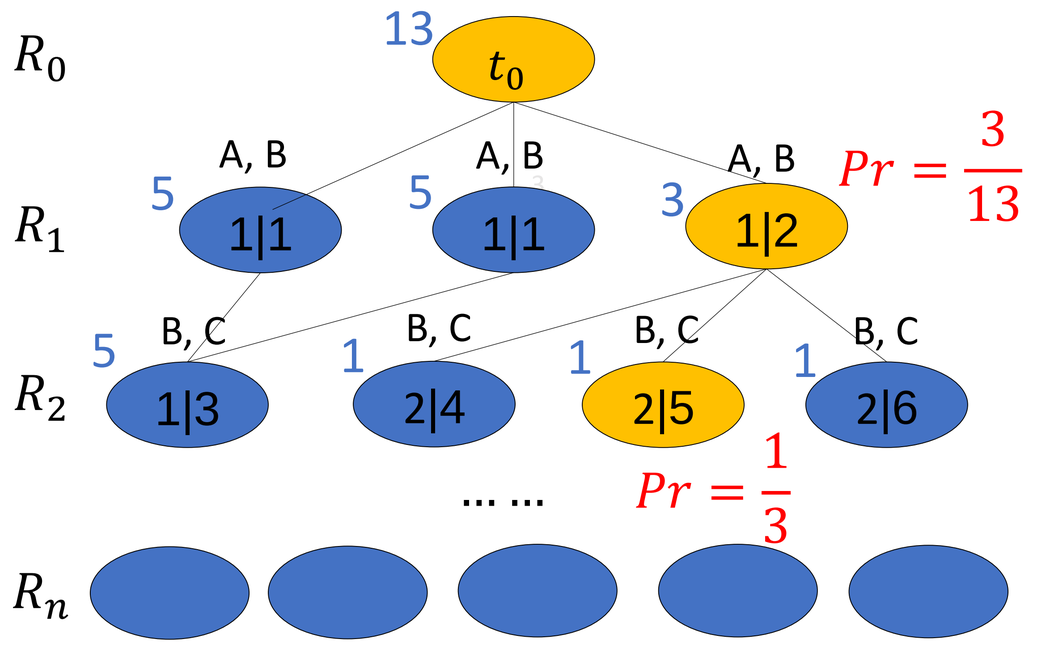


Example: a chain join $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$

When exact weights are known

for each $0 \le i \le n-1$,

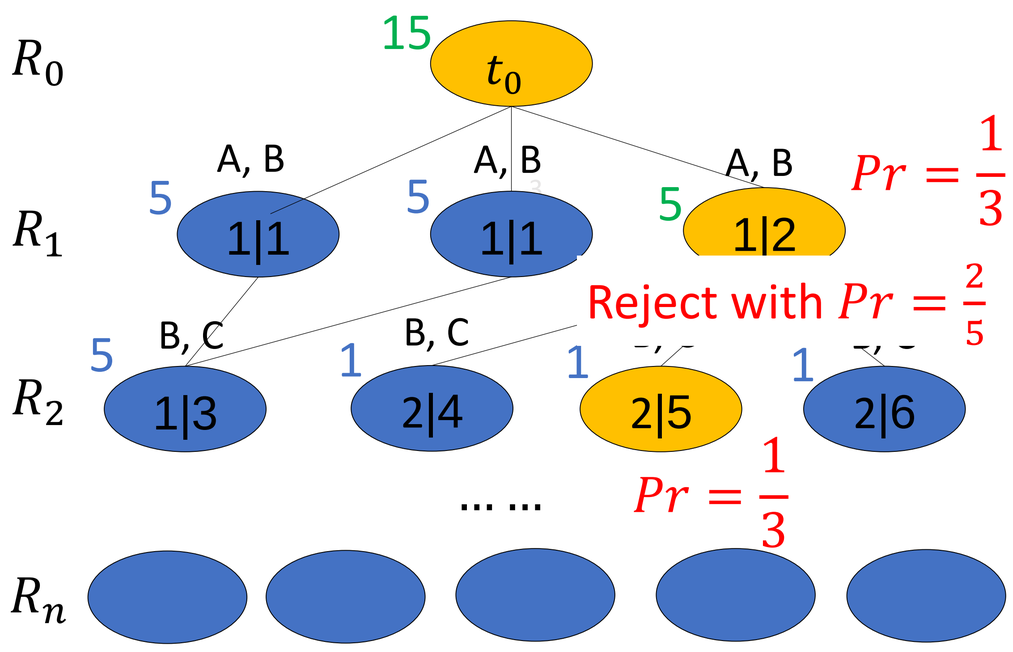
sample $t_{i+1} \in t_i \bowtie R_i$ with probability $\propto \frac{w(t_{i+1})}{w(t_i)}$



Chaudhuri el al.'s algorithm is a special case when n = 2.

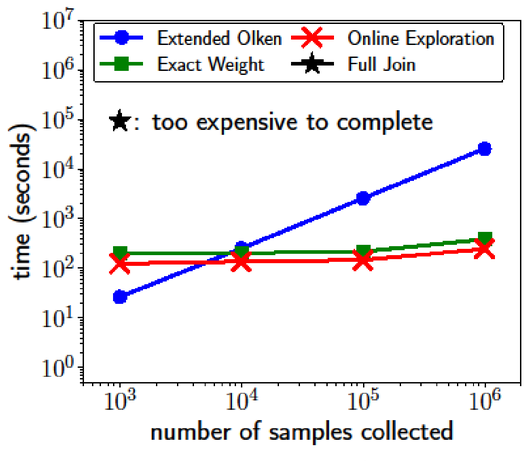
When exact weights are unknown

- Use an upper bound of weight $W(t)$ instead
  - Additional constraints apply (see paper Sec. 3)
  - Reject with prob. $\frac{W(t)-\sum_{t' \in ch(t)} W(t')}{W(t)}$ at each step

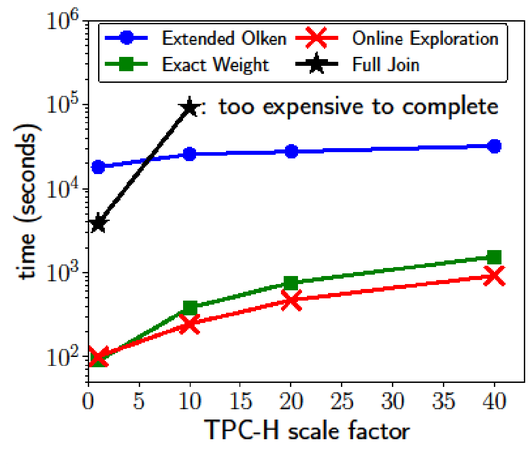


- Different instantiation of $W(t)$ => different sampling algorithms

Evaluation



7-table cyclic join on TPC-H (scale factor 10)



Scalability: time to collect 10^6 samples

Extended Olken: an extension to Olken's algorithm
Online Exploration: using Wander Join to estimate upper bounds
Exact Weight: the case when weights are known (extension to Chaudhuri et al.'s algorithm)
Full Join: hash join