OUTPUT-SENSITIVE CONJUNCTIVE QUERY EVALUATION

Shaleen Deep[1], Hangdong Zhao[2], Austen Z. Fan[2], Paris Koutris[2]

[1]Microsoft GSL, [2]University of Wisconsin-Madison

INTRODUCTION

Join query evaluation is one of the most important operation performed by DBMS (both graph and relational)

Staggering number of join algorithms and variants have been developed over the last 50 years

Lots of practical optimizations (such as bloom filters, predicate transfer, etc.) have been integrated into evaluation engines to speed up evaluation

Question: What is the optimal time complexity of join query evaluation?

INTRODUCTION

Parameters for expressing evaluation time complexity

- Database D
- Join query Q
- Output size $|OUT| = |Q(D)|$

Time complexity of evaluating $Q(D) = O(|D| + g(|D|^{x}, |OUT|^{y}) + |OUT|)$

WHAT IS THE FUNCTION G(.,.) AND WHAT IS THE VALUE OF EXPONENTS **X** AND **Y**?

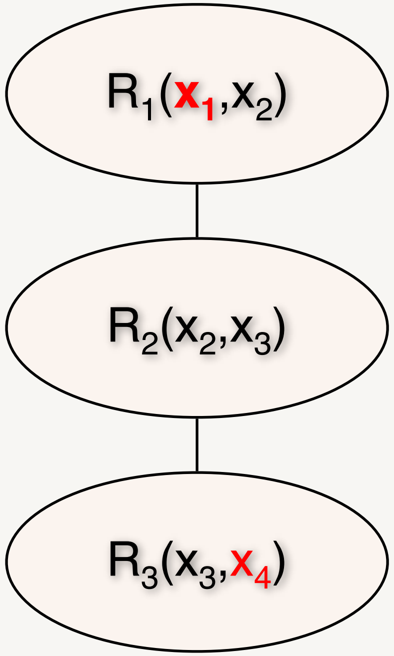
PRELIMINARIES

A join query is of the form

$$Q(x_1,x_2,\ldots,x_k) = R_1(\mathbf{y_1}) \bowtie R_2(\mathbf{y_2}) \bowtie \ldots \bowtie R_n(\mathbf{y_n})$$

Consider the 3-path query: $Q(x_1,x_4) = R_1(x_1,x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_4)$

- Acyclic queries can be visualized via a *join tree*
- Each node in the tree corresponds to a relation
- variables in the tree form a connected structure



YANNAKAKIS ALGORITHM

Theorem 1 [VLDB 1981]. Given any acyclic CQ Q and a database D, Q(D) can be evaluated in time O(|D| + |D|•|OUT| + |OUT|).

Properties:

1. Yannakakis algorithm gives the running time guarantee for *any* join tree!
2. The algorithm is output-sensitive

OUR MAIN IDEA: CLEVERLY PARTITION THE INPUT DATA AND USE DIFFERENT JOIN TREES FOR DIFFERENT PARTITIONS

OUR CONTRIBUTION

We present a novel algorithm that improves upon the Yannakakis algorithm. In particular, we show that it is possible to evaluate an acyclic query Q on database D in time where f(Q) > 1

This is the first improvement of the Yannakakis algorithm in over 40 years using *combinatorial* algorithms

We show that subject to popular conjectures, our algorithm is optimal for a large class of queries

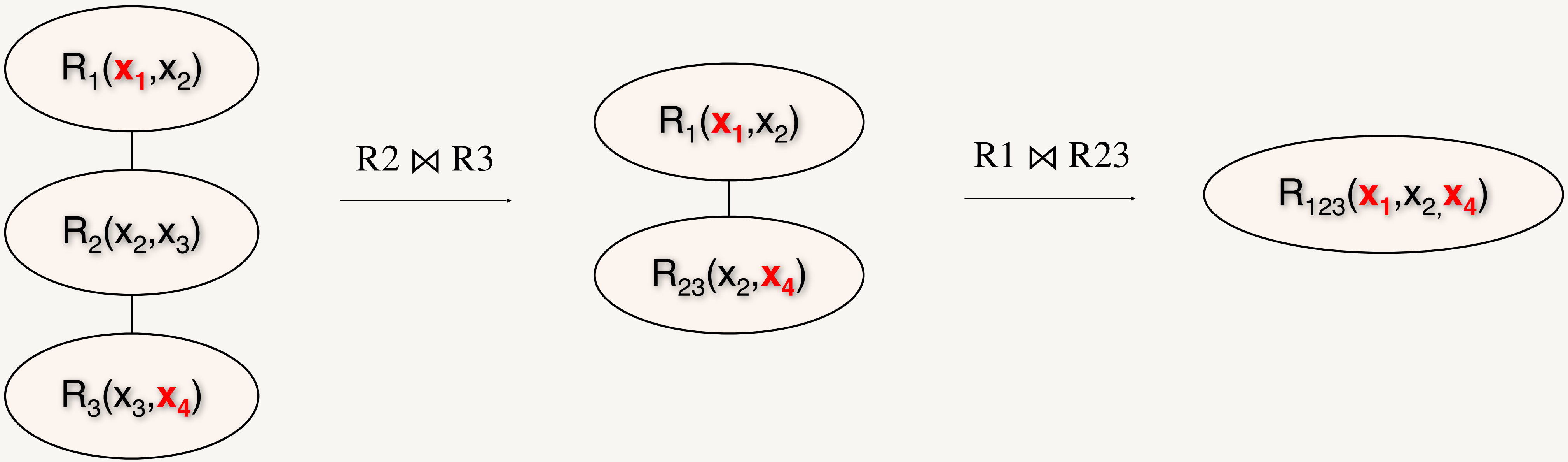
YANNAKAKIS ALGORITHM

Step 1. Pick any join tree and remove all "dangling tuples"

Step 2. Process nodes in bottom-up fashion

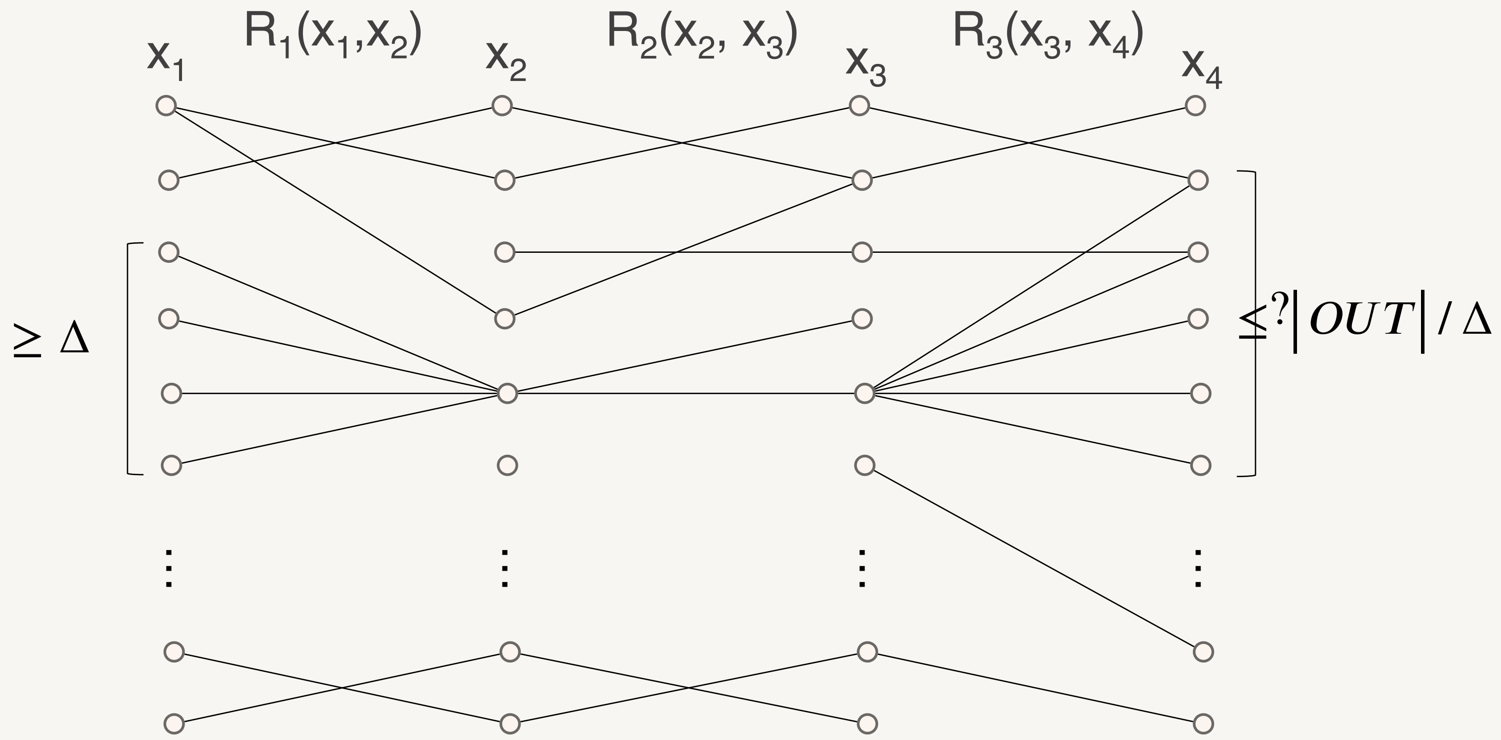
Step 2.1 Join relation with all relations in its subtree

Step 2.2 Project on the output variables in the subtree and the join variables



KEY IDEAS

$$Q(x_1,x_4) = R_1(x_1,x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_4)$$

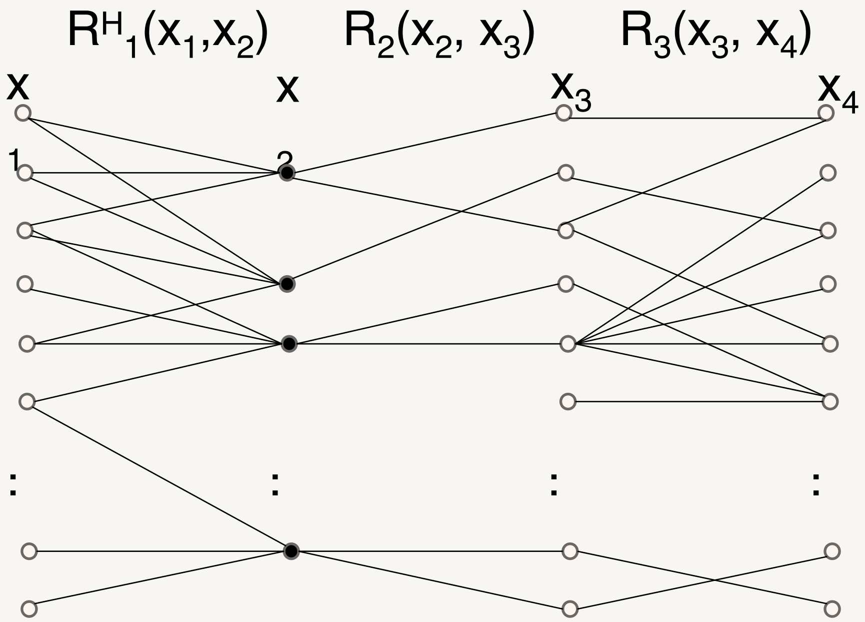


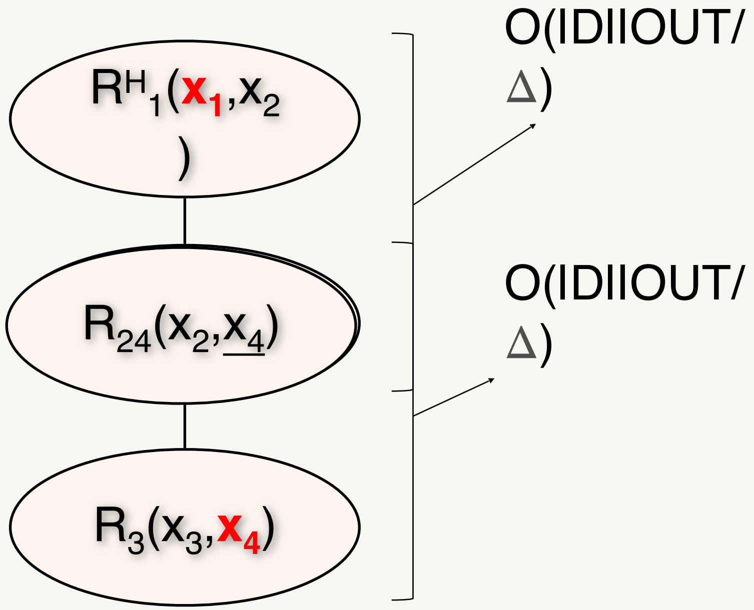
KEY IDEAS

Pick a relation that contains an output variable (say $R_1(x_1,x_2)$)

Filter rows of $R_1(x_1,x_2)$: create relation $R^H_1(x_1,x_2)$ where $\deg(x_2, R_1) > \Delta$ (aka the *heavy* part)

$$Q^H(x_1,x_4) = R^H_1(x_1,x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_4)$$

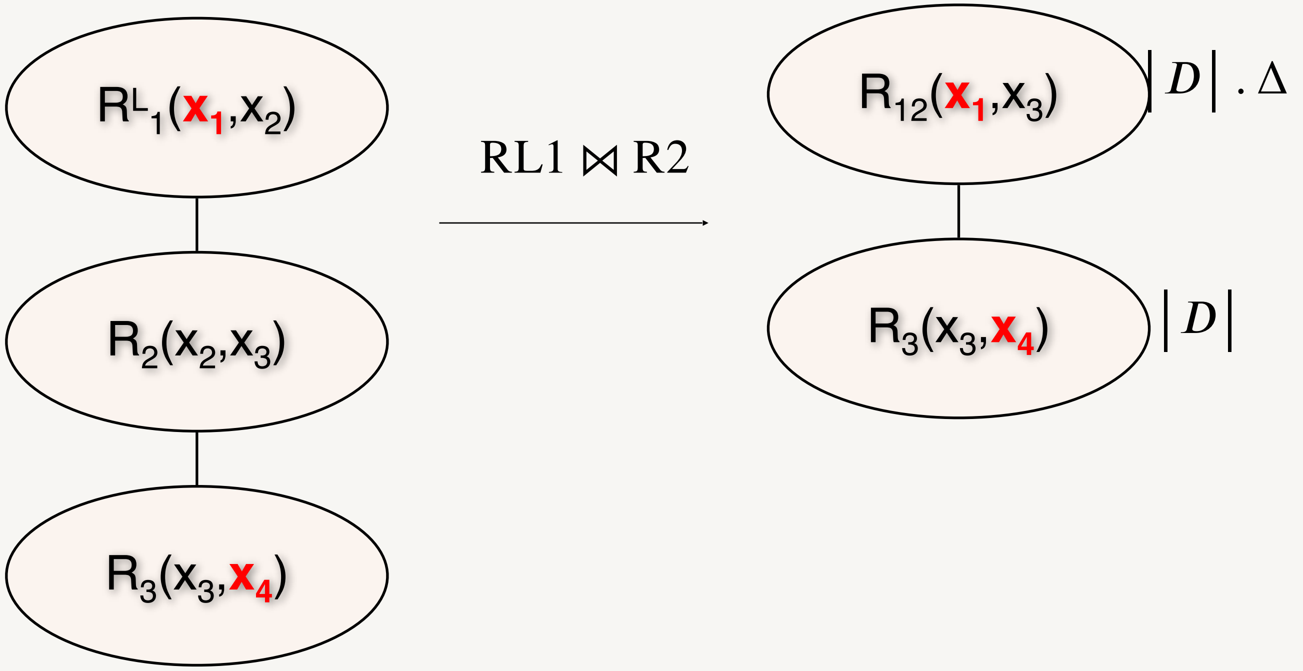




KEY IDEAS

Next, we process the light part $R^L_1(x_1,x_2) = R_1 \setminus R^H_1(x_1,x_2)$

$$Q^L(x_1,x_4) = R^L_1(x_1,x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_4)$$



Repeat the same idea on the new query!

FINAL ALGORITHM

In each iteration, we reduce the number nodes in the join tree by one.

For iteration i,

- Processing of the heavy partition requires O(|D||OUT|/Δ)
- Processing of the light partition requires O(|D|Δi)

Assuming k relations in the query, the running time is minimized when

$$|D||OUT|/\Delta = |D|$$

Plugging in the optimal value of ,

Total running time = O( )

EXTENSIONS

Our framework can also be extended to queries involving GROUP BY queries and aggregations

For queries that are cyclic, we can apply our results by first converting the cyclic query into acyclic schema using the standard idea of “tree decompositions”

LOWER BOUNDS

Boolean k-clique conjecture: There is no real $\epsilon > 0$ such that computing the k-clique problem (with $k \geq 3$) over the Boolean semiring in an (undirected) n-node graph requires time using a combinatorial algorithm

Theorem: There exists a query Q with l output variables such that no combinatorial algorithm can compute Q(D) in time subject to the Boolean k-clique conjecture for any real $\epsilon > 0$

CONCLUSION AND FUTURE WORK

- In this talk, we present a novel algorithm that improves upon the Yannakakis algorithm.

- Future Work 1: Practical implementation of our work!
  - The algorithm is a join-project plan and thus, can be readily implemented via SQL queries

- Future Work 2: Discover more cool algorithms!
  - Can the ideas be extended to other join queries (such as band joins)?
  - Algorithms that consider other parameters such as minimizing number of semijoins