CS848 Fall 2025: Algorithmic Aspects of Query Processing

Worst-case Optimal Joins

Xiao Hu

Sep 15, 2025

Agenda

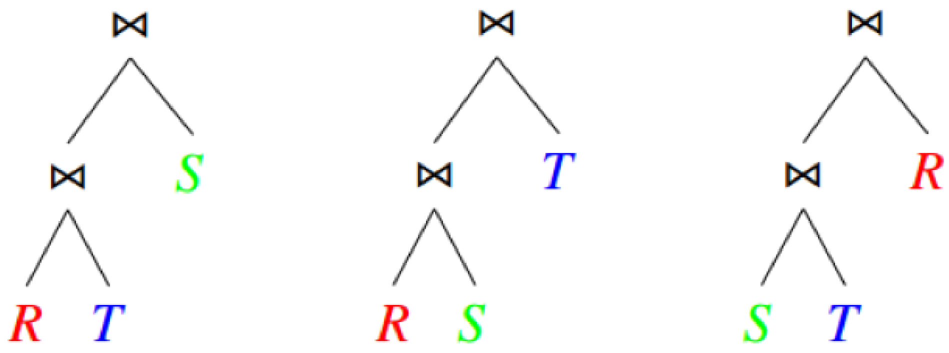
- Last class: Traditional query processing
- This class: Worst-case optimal join algorithms
  - Limitations of Pairwise Framework
  - AGM bound
  - Worst-case Optimal Join Algorithms
  - Applications

Related Pointers

- Skew strikes back: New Developments in the Theory of Join Algorithms. SIGMOD Record 2013.
- A. ATSERIAS, M. GROHE and D. MARX, "Size bounds and query plans for relational joins," FOCS 2008.
- S. ABITEBOUL, R. HULL and V. VIANU, "Foundations of Databases."
- M. YANNAKAKIS, "Algorithms for acyclic database schemes," VLDB 1981.
- G. GOTTLOB, N. LEONE and F. SCARCELLO, "Hypertree Decompositions and Tractable Queries," Journal of Computer and System Sciences 64 (2002) .
- M. GROHE, T. SCHWENTICK and L. SEGOUFIN, "When is the evaluation of conjunctive queries tractable ?," STOC 2001 .
- G. GOTTLOB, G. GRECO and F. SCARCELLO, "Treewidth and Hypertree Width".

Recap on Pairwise Framework

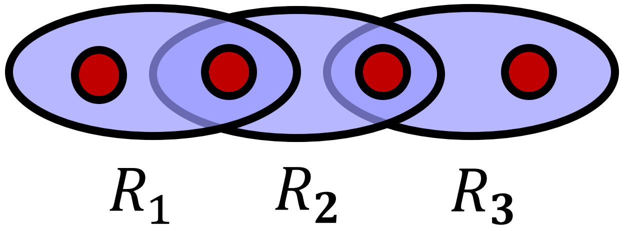
- **Join Query:** A highly optimized version of Pairwise Framework
  - A join plan is a binary tree
  - Estimate the cost of each query plan using data statistics
  - Pick the one with the minimum cost

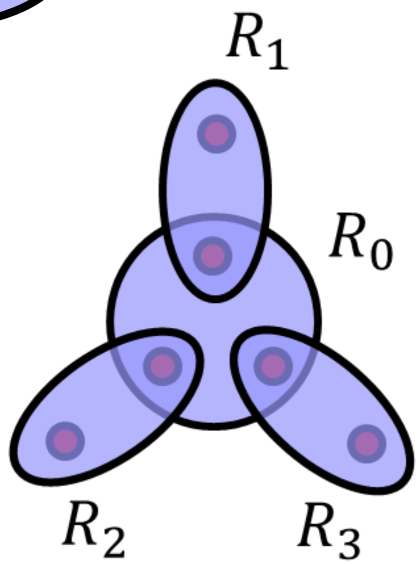


$$Q_{\Delta} := R(A,B) \bowtie S(B,C) \bowtie T(C,A)$$

Recap on Yannakakis algorithm and Acyclic Joins

- Any acyclic join can be computed efficiently!

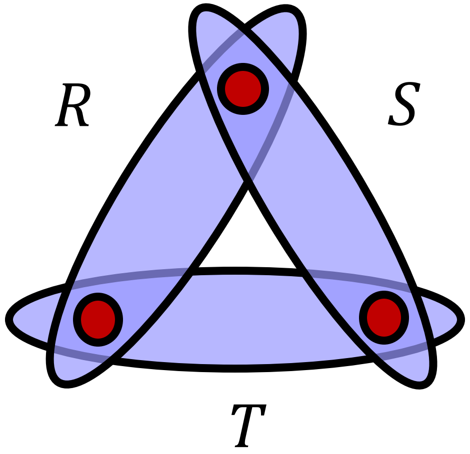




  - Path join: $R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$
  - Star join : $R_0(A,B,C) \bowtie R_1(A,D_1) \bowtie R_2(B,D_2) \bowtie R_3(C,D_3)$

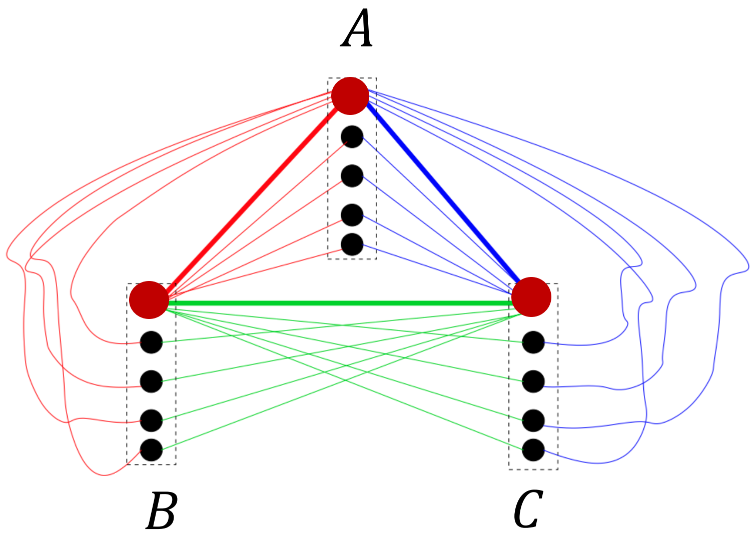
- No optimality on triangle join: $R(A,B) \bowtie S(B,C) \bowtie T(A,C)$

- How worse could Yannakakis algorithm be?



Pathological instance for Yannakakis algorithm

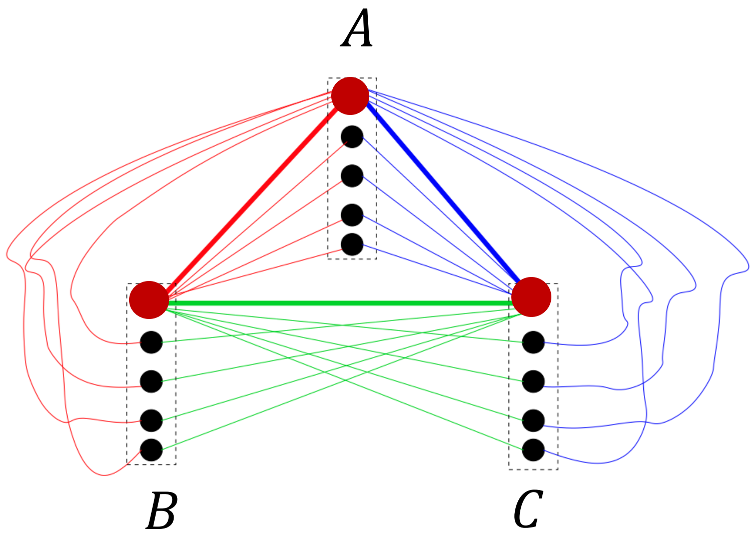
- No more tuples can be removed
- The number of input tuples is $6n + 3$
- The number of triangle join results is $3n + 1$
- But any pairwise join would generate n^2 intermediate results



$$Q_\Delta := R(A,B) \bowtie S(B,C) \bowtie T(C,A)$$

$$R(A,B) = \{(a_0, b_i): i \in [n]\} \cup \{(a_i, b_0): i \in [n]\} \cup \{(a_0, b_0)\}$$
$$S(B,C) = \{(b_0, c_i): i \in [n]\} \cup \{(b_i, c_0): i \in [n]\} \cup \{(b_0, c_0)\}$$
$$T(A,C) = \{(a_0, c_i): i \in [n]\} \cup \{(a_i, c_0): i \in [n]\} \cup \{(a_0, c_0)\}$$

Algorithm 1: The Power of Two Choices



- Consider each value $a \in (\pi_A R) \cap (\pi_A T)$:

$$R(a, B) \bowtie S(B, C) \bowtie T(C, a)$$
$$\Leftrightarrow \left((\pi_B \sigma_{A=a} R) \times (\pi_C \sigma_{A=a} T)\right) \cap S$$

  – Choice 1: for each "neighbor" b, and for each "neighbor" c, check if $(b, c) \in S$

  – Choice 2: for each $(b, c) \in S$, check if b is "neighbor" of a and c is "neighbor" of a

$$Q_\Delta := R(A, B) \bowtie S(B, C) \bowtie T(C, A)$$

Algorithm 1: The Power of Two Choices

- Idea: Make an individual choice for each value $a \in (\pi_A R) \cap (\pi_A T)$

- How to make a choice?
  - Always choose the "cheaper" one!
  - Choice 1: for each "neighbor" b, and for each "neighbor" c, check if $(b, c) \in S$
  - Choice 2: for each $(b, c) \in S$, check if b is "neighbor" of a and c is "neighbor" of a

For value a with $|\pi_B \sigma_{A=a} R| \cdot |\pi_C \sigma_{A=a} T| < |S|$

For value a with $|\pi_B \sigma_{A=a} R| \cdot |\pi_C \sigma_{A=a} T| \geq |S|$

Algorithm 1: The Power of Two Choices

- Hashing Indexes
- Analysis: $O\left(\sqrt{|R| \cdot |S| \cdot |T|} + |R| + |S| + |T|\right)$
  - $O(N^{1.5})$ if $|R| = |S| = |T| = N$

Algorithm 2: The delay of Computation

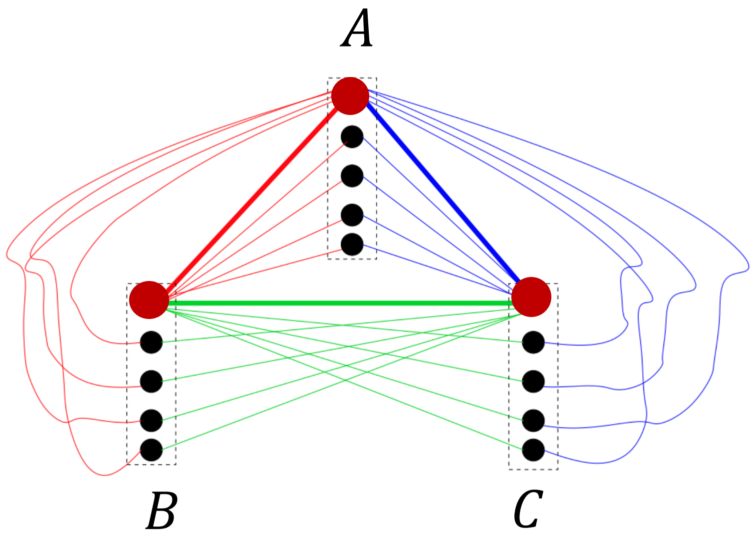
- Consider each value $a \in (\pi_A R) \cap (\pi_A T)$:

  $$R(a, B) \bowtie S(B, C) \bowtie T(C, a)$$

  - Consider each value $b \in (\pi_B \sigma_{A=a} R) \cap (\pi_B S)$

    $$R(a, b) \bowtie S(b, C) \bowtie T(C, a)$$

    - Consider each value $c \in (\pi_C \sigma_{B=b} S) \cap (\pi_C \sigma_{A=a} T)$, and output (a, b, c)



$$Q_\Delta := R(A, B) \bowtie S(B, C) \bowtie T(C, A)$$

Algorithm 2: The delay of Computation

- Hashing Indexes
- Analysis: $O\left(\sqrt{|R| \cdot |S| \cdot |T|} + |R| + |S| + |T|\right)$
  - $O(N^{1.5})$ if $|R| = |S| = |T| = N$

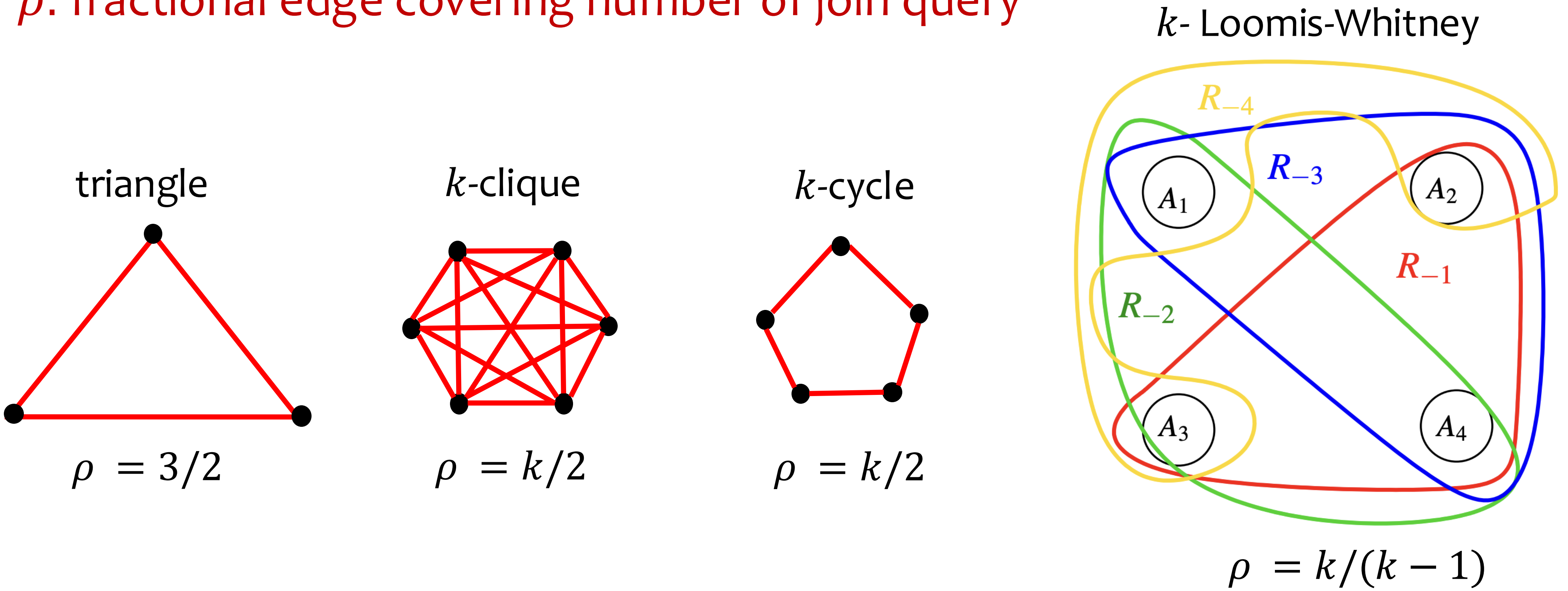
Worst-case optimality for Triangle Join

- Consider a hard instance of the triangle join where
  - $|A| = |B| = |C| = \sqrt{N}$
  - $R(A, B)$ is Cartesian product between A and B
  - $S(B, C)$ is Cartesian product between B and C
  - $T(C, A)$ is Cartesian product between A and C

- Input size is N and output size is $N^{\frac{3}{2}}$
- So, any algorithm needs to spend $\Omega\left(N^{\frac{3}{2}}\right)$ time to compute this instance

AGM bound

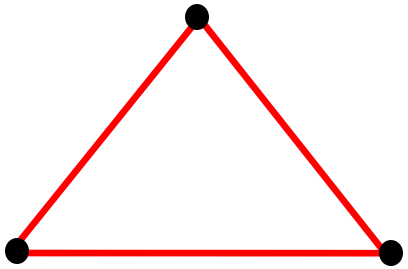
- For a join query Q, any database of input size N can produce at most $O(N^{\rho})$ join results
  - ρ: fractional edge covering number of join query



AGM bound is tight

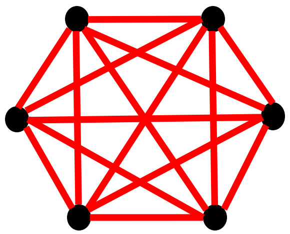
- A hard instance: For a join query Q, and parameter N, there always exists a database of input size N can produce $\Omega(N^{\rho})$ join results
- Duality between fractional edge covering and fractional vertex packing

triangle



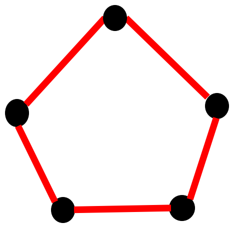
$\tau = 3/2$

k-clique



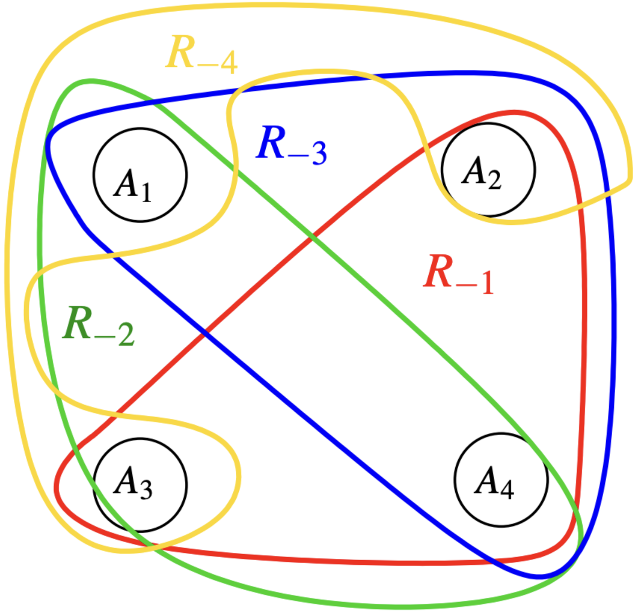
$\tau = k/2$

k-cycle



$\tau = k/2$

k- Loomis-Whitney



$\tau = k/(k-1)$

Next Class

- How to prove AGM bound? – Many different ways!
- Can we design an algorithm whose running time matches the AGM bound?
- Can we apply the WCOJ algorithm to derive an output-sensitive algorithm for cyclic joins?