# CS848 Fall 2025: Algorithmic Aspects of Query Processing

# Output-Optimal Algorithms for Join-Aggregate Queries

Xiao Hu

Sep 24, 2025

# Agenda

- Last class: Worst-case optimal join algorithms
- This class: join-aggregate queries
  - Matrix multiplication and its Variant
  - Limitations of Yannakakis algorithm
  - Output-optimal algorithm for Chain Matrix Multiplication
  - General join-aggregate queries
  - General Algorithm

# Related Pointers

- X. HU, “Output-optimal Algorithms for Join-Aggregate Queries,” PODS 2025.
- 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”.

# (Natural) Join Query

- A join query $q \coloneqq R_1(e_1) \bowtie R_2(e_2) \bowtie \cdots \bowtie R_k(e_k)$
  - $e_1, e_2, \cdots, e_k$ are subsets of attributes
  - $R_1, R_2, \cdots, R_k$ are relations
  - $q = \{t \in \mathrm{dom}(e_1 \cup e_2 \cup \cdots \cup e_k) : \forall i \in [k], \pi_{e_i} t \in R_i\}$

- Examples in graphs:
  - Listing triangles: $E_1(A, B) \bowtie E_2(B, C) \bowtie E_3(A, C)$
  - Listing length-$k$ chains: $E_1(A_1, A_2) \bowtie E_2(A_2, A_3) \bowtie \cdots \bowtie E_k(A_k, A_{k+1})$
  - Listing $k$-way stars: $E_1(A_1, B) \bowtie E_2(A_2, B) \bowtie \cdots \bowtie E_k(A_k, B)$
  - Listing length-4 cycles: $E_1(A, B) \bowtie E_2(B, C) \bowtie E_3(C, D) \bowtie E_4(D, A)$

# Commutative Semi-ring and Ring

- A commutative semi-ring $(\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$
  - $(\mathbf{D}, \oplus, \mathbf{0})$ is a commutative monoid with identity $\mathbf{0}$
    - $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
    - $a \oplus b = b \oplus a$
    - $a \oplus \mathbf{0} = \mathbf{0} \oplus a = a$
  - $(\mathbf{D}, \otimes, \mathbf{1})$ is a commutative monoid with identity $\mathbf{1}$
    - $(a \otimes b) \otimes c = a \otimes (b \otimes c)$
    - $a \otimes b = b \otimes a$
    - $a \otimes \mathbf{1} = \mathbf{1} \otimes a = a$
  - $\otimes$ distributes over $\oplus$
    - $(a \otimes b) \oplus (a \otimes c) = a \otimes (b \oplus c)$
  - $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$ for any element $a \in \mathbf{D}$

- Additional condition for ring:
  - $(\mathbf{D}, \oplus, \mathbf{0})$ is a group
    - each element $a \in \mathbf{D}$ has an additive inverse $-a$: $a \oplus (-a) = \mathbf{0}$

# Join-Aggregate Query = Aggregation over Join Query

- A join-aggregate query under $(\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$

$$Q(\mathrm{y}) =: \bigoplus_{\bar{\mathrm{y}}} q = \bigoplus_{\bar{\mathrm{y}}} R_1(e_1) \bowtie R_2(e_2) \bowtie \cdots \bowtie R_k(e_k)$$

  - $(\mathrm{y}, \bar{\mathrm{y}})$ is a partition of all attributes $e_1 \cup e_2 \cup \cdots \cup e_k$
  - A full join if $\mathrm{y} = e_1 \cup e_2 \cup \cdots \cup e_k$
  - Each tuple $t$ is annotated with $\delta t \in \mathbf{D}$
  - The annotation of a join result $t$ is $\delta t = (\delta\pi_{e_1} t) \otimes (\delta\pi_{e_2} t) \otimes \cdots \otimes (\delta\pi_{e_k} t)$
  - $Q(\mathrm{y}) = \left\{(t', \delta t') \in (\pi_{\mathrm{y}} q) \times \mathbf{D} : \delta t' = \bigoplus_{t \in q : \pi_{\mathrm{y}} t = t'} \delta t\right\}$

# Example of Join-Aggregate Queries on $(\mathbb{Z},+,\times,0,1)$

$$\begin{bmatrix}0&0&0&1\\1&0&4&0\\3&0&0&0\\0&1&0&0\end{bmatrix}\times\begin{bmatrix}3&2&0&0\\0&3&0&1\\0&5&0&0\\0&0&1&0\end{bmatrix}=\begin{bmatrix}0&0&1&0\\3&22&0&0\\9&6&0&0\\0&3&0&1\end{bmatrix}$$

$R_1(A,B)$

| A | B | $\delta(\cdot)$ |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 1 | 1 |
| 2 | 3 | 4 |
| 3 | 1 | 3 |
| 4 | 2 | 1 |

$R_2(B,C)$

| B | C | $\delta(\cdot)$ |
|---|---|---|
| 1 | 1 | 3 |
| 1 | 2 | 2 |
| 2 | 2 | 3 |
| 2 | 4 | 1 |
| 3 | 2 | 5 |
| 4 | 3 | 1 |

$R_1 \bowtie R_2$

| A | B | C | $\delta(\cdot)$ |
|---|---|---|---|
| 1 | 4 | 3 | $1\cdot 1=1$ |
| 2 | 1 | 1 | $1\cdot 3=3$ |
| 2 | 1 | 2 | $1\cdot 2=2$ |
| 2 | 3 | 2 | $4\cdot 5=20$ |
| 3 | 1 | 1 | $3\cdot 3=9$ |
| 3 | 1 | 2 | $3\cdot 2=6$ |
| 4 | 2 | 2 | $1\cdot 3=3$ |
| 4 | 2 | 4 | $1\cdot 1=1$ |

$\sum_B R_1 \bowtie R_2$

| A | C | $\delta(\cdot)$ |
|---|---|---|
| 1 | 3 | 1 |
| 2 | 1 | 3 |
| 2 | 2 | $2+20=22$ |
| 3 | 1 | 9 |
| 3 | 2 | 6 |
| 4 | 2 | 3 |
| 4 | 4 | 1 |

# Examples of Commutative Semi-ring

| D | $\oplus$ | $\otimes$ | 0 | 1 | Name |
|---|---|---|---|---|---|
| {true, false} | ∨ | ∧ | false | true | Boolean |
| $\mathbb{N}$ | $+$ | $\times$ | $0$ | $1$ | Natural sum-product |
| $\mathbb{R}/\mathbb{Z}$ | $+$ | $\times$ | $0$ | $1$ | Real/Integer sum-product |
| $(-\infty, +\infty]$ | min | $+$ | $+\infty$ | $0$ | Min-sum |
| $[-\infty, +\infty)$ | max | $+$ | $-\infty$ | $0$ | Max-sum |
| $(0, +\infty]$ | min | $\times$ | $+\infty$ | $1$ | Min-product |
| $[0, +\infty)$ | max | $\times$ | $0$ | $1$ | Max-product |
| $\mathbb{N}[\mathbf{X}]$ | $+$ | $\times$ | $0$ | $1$ | Polynomials over **X** |

Boolean: conjunctive query

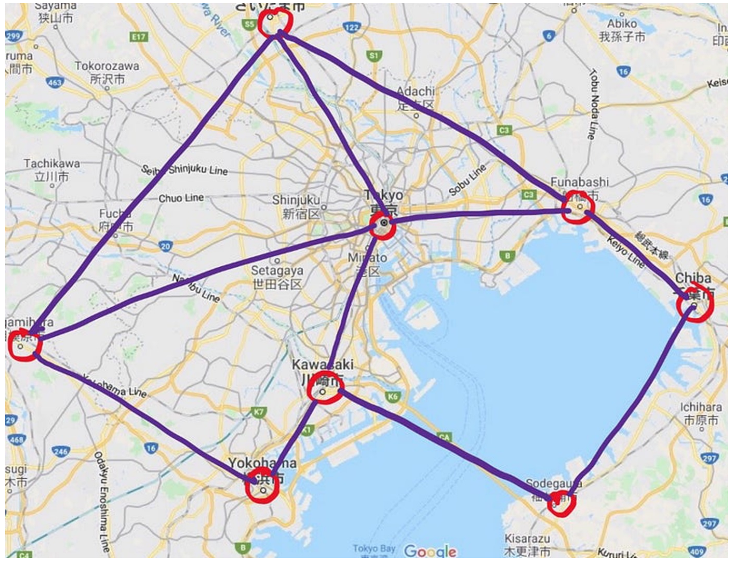
Sum-product: counting query; inference in probabilistic graphical models; matrix multiplication; permanent; discrete fourier transformation

Min-sum: shortest path

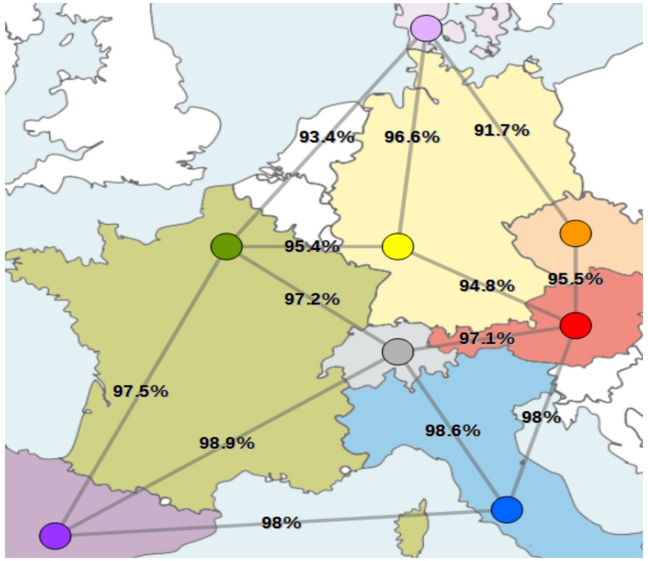
Max-product: maximum posteriori in probabilistic graphical models; maximum likelihood decoder for linear codes

Polynomials: data provenance

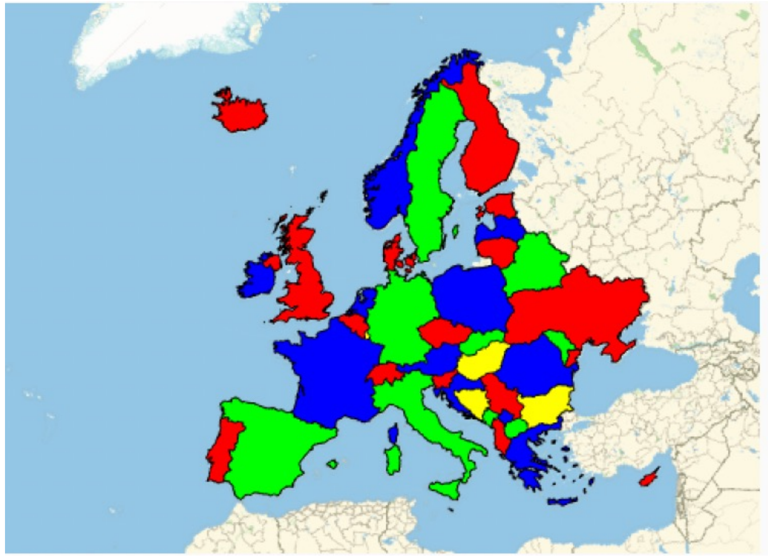
# Join-Aggregate Queries in Disguise



shortest path




maximum reliability



colorability

# What is lower and upper bound for acyclic join-aggregate queries by semi-ring algorithms?

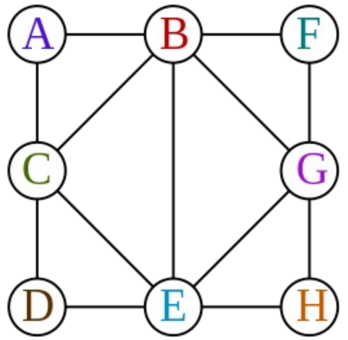
Answer: $\Theta\left(N \cdot OUT^{1-\frac{1}{\text{fnfhtw}}} + OUT\right)$ [H25]

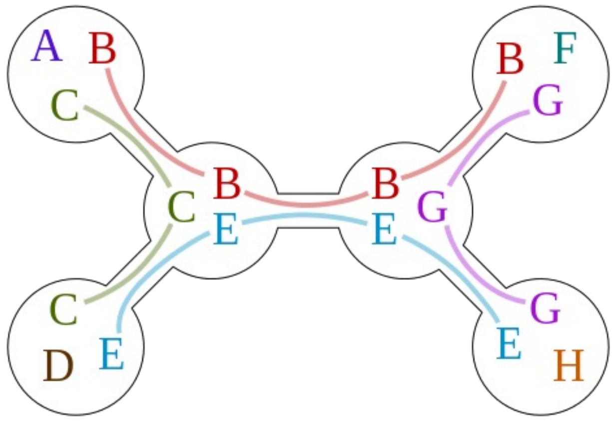
where fnfhtw is the free-connex fractional hypertree width of the query

$\text{fnfhtw} = 1$ for free-connex queries;
$\text{fnfhtw} = k$ for star queries with $k$ relations;
$\text{fnfhtw} = 2$ for chain queries with arbitrary relations

# (Recap) Tree Decomposition

- For a join-project query $\pi_y Q$ with $Q = (V, E)$, a tree decomposition for $Q$ is a tree $T$ with the set of nodes $V_T$ and a mapping $\lambda: V_T \to 2^V$ such that
  - (coverage) for each relation $e \in E$, there exists a node $u \in T$ with e $\subseteq \lambda_u$
  - (connectness) for each attribute $x \in V$, the set of nodes containing $x$, i.e., $\{u \in V_T: x \in \lambda_u\}$ forms a connected subtree of $T$

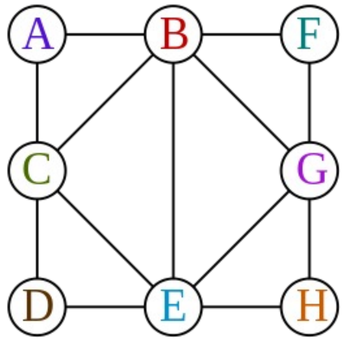


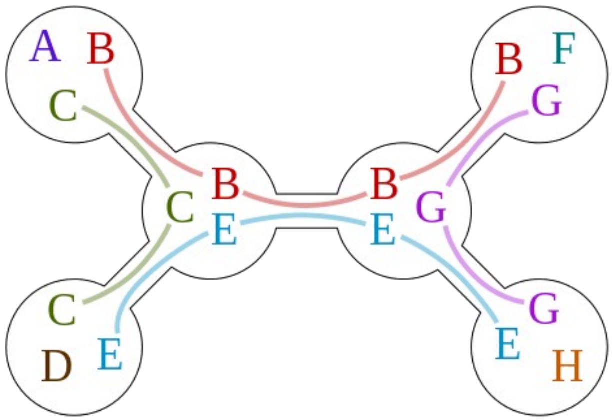



# Free-Connex Tree Decomposition

- For a join-project query $\pi_y Q$ with $Q = (V, E)$, a free-connex tree decomposition for $Q$ is a tree $T$ with the set of nodes $V_T$ and a mapping $\lambda: V_T \rightarrow 2^V$ such that
  - (coverage) for each relation $e \in E$, there exists a node $u \in T$ with e $\subseteq \lambda_u$
  - (connectness) for each attribute $x \in V$, the set of nodes containing $x$, i.e., $\{u \in V_T : x \in \lambda_u\}$ forms a connected subtree of $T$
  - (connex) there exists a connected subtree $T_{\text{con}} \subseteq T$ containing the root node $r$ of $T$ and the union of nodes in $T_{\text{con}}$ is exactly the output attributes
- The sub-join query induced by node $u$ is

$$Q_u = (\lambda_u, \{u \in V_T : e \cap u \neq \emptyset\}).$$



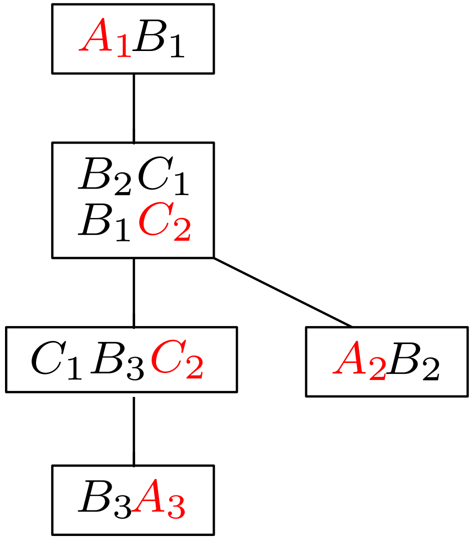

Is this a free-connex tree decomposition of $\pi_{A,B,C,D,E} Q$?

Is this a free-connex tree decomposition of $\pi_{A,B,C,F} Q$?

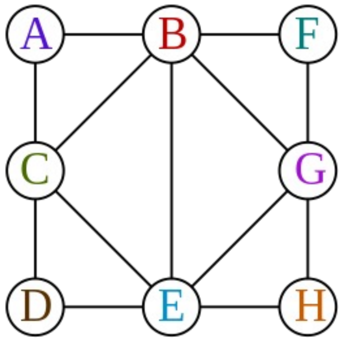
# (Recap) Fractional Hypertree Width

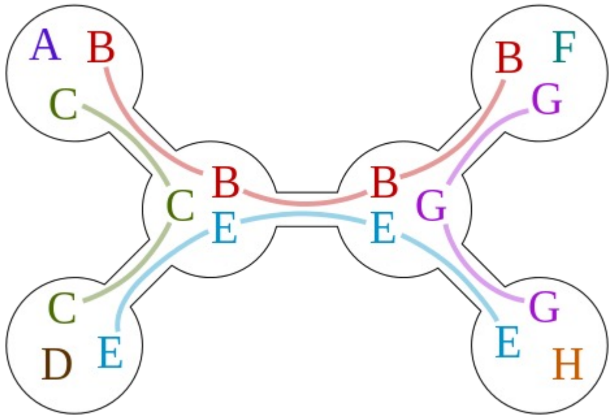
$$\text{fhtw} = \min_{\text{tree decomposition } T} \; \max_{\text{node } u \in T} \; \rho\,(Q_u)$$

$\pi_{A_1,A_2,A_3,C_2,C_3}$
$R_1(A_1, B_1)$
$\bowtie R_2(A_2, B_2)$
$\bowtie R_3(A_3, B_3)$
$\bowtie R_4(B_1, B_2, C_1, C_2)$
$\bowtie R_5(C_1, B_3, B_4, C_2)$




fhtw = 1



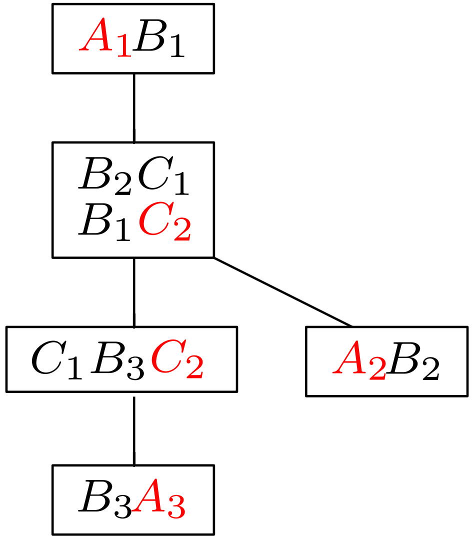

fhtw = 1.5

# Free-connex Fractional Hypertree Width

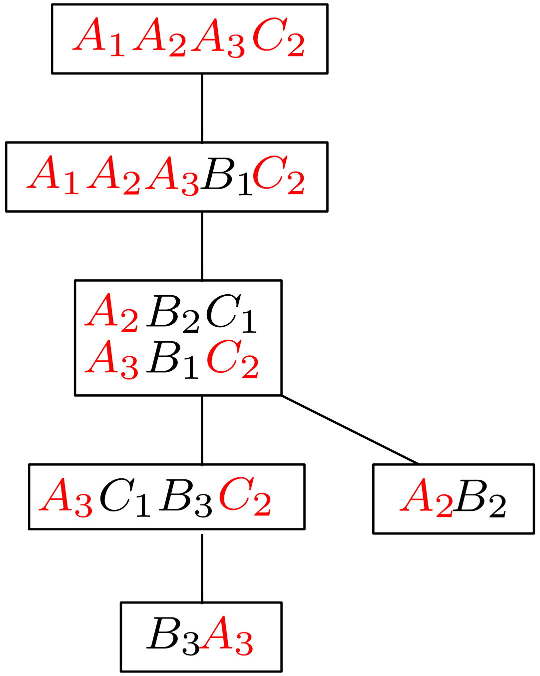
$$\text{fnfhtw} = \min_{\substack{\text{free-connex} \\ \text{tree decomposition } T}} \max_{\text{node } u \in T} \rho^*(Q_u)$$

$\pi_{A_1,A_2,A_3,C_2,C_3}$
$R_1(A_1, B)$
$\bowtie R_2(A_2, B)$
$\bowtie R_3(A_3, B)$
$\bowtie R_4(B_1, B_2, C_1, C_2)$
$\bowtie R_5(C_1, B_3, C_2)$

fnfhtw = 4



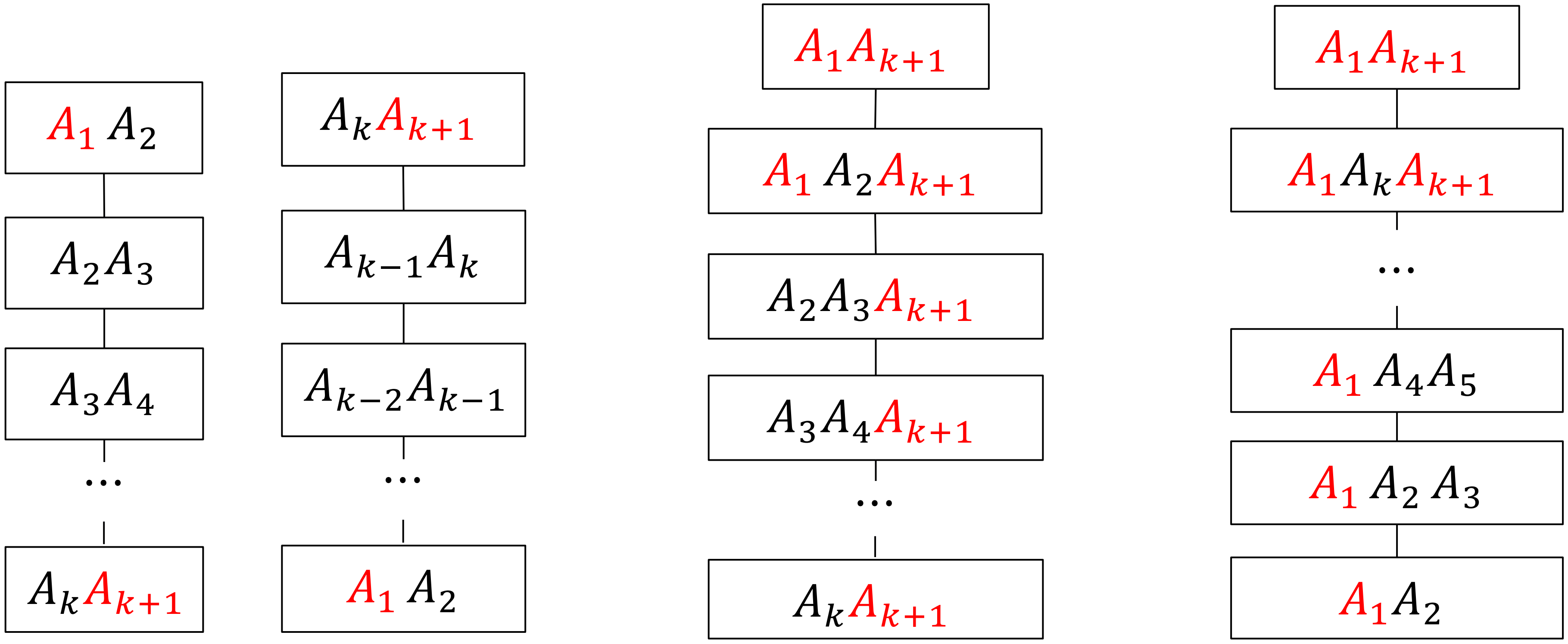

a tree decomposition but not free-connex




a free-connex tree decomposition

# fnfhtw = 2 for Chain MM

$$Q(A_1, A_{k+1}) \coloneqq R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$$




tree decompositions but not free-connex

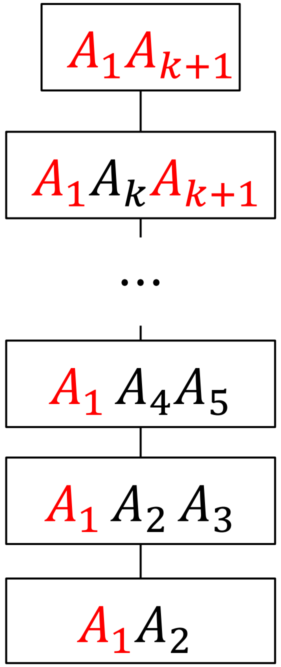
free-connex tree decompositions

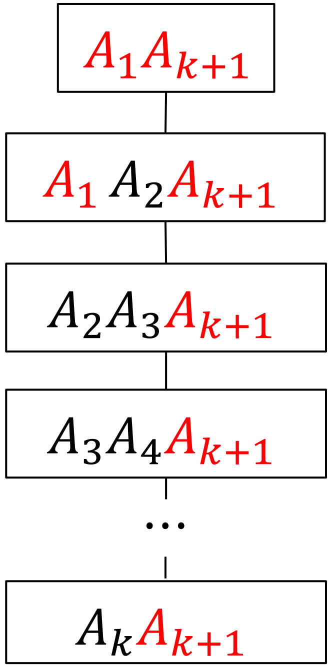
# A Free-Connex Tree Decomposition is A Query Plan

$$Q(A_1, A_{k+1}) \coloneqq R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$$

$R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$ →

← $R_1(A_1, A_2) \bowtie R_2(A_2, A_3) \bowtie \cdots \bowtie R_k(A_k, A_{k+1})$

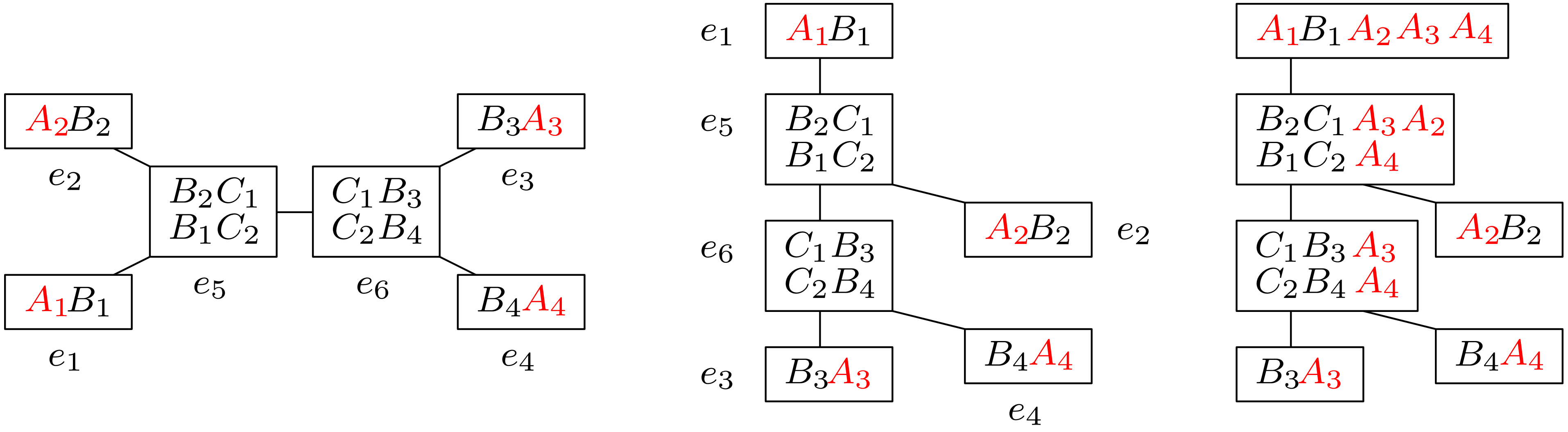


# Hybrid Yannakakis Algorithm

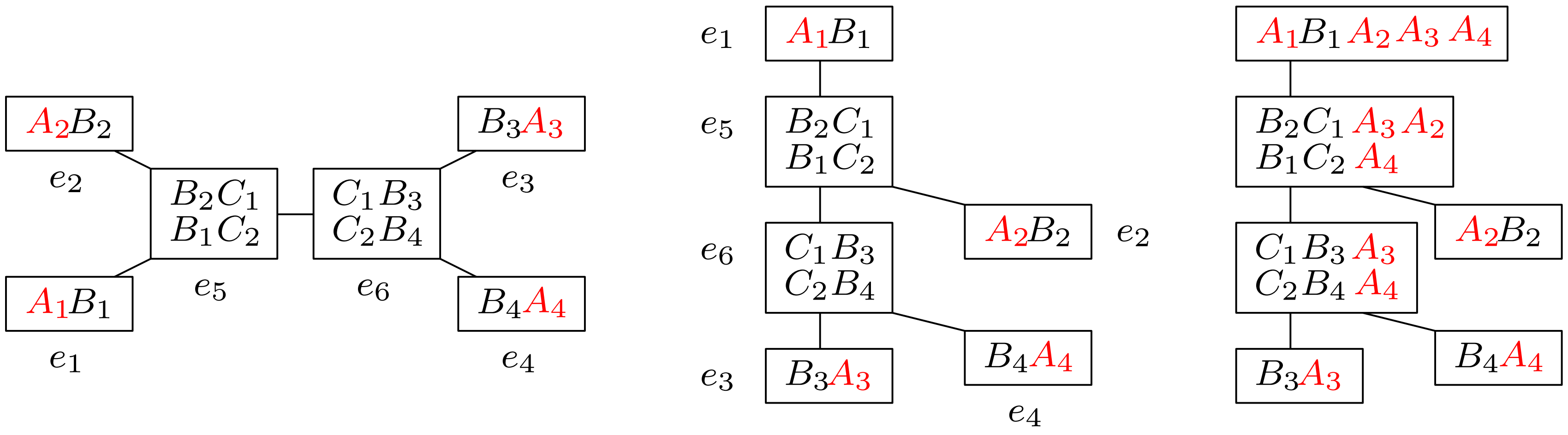
- Partition the input instance and Find a good query plan for each sub-instance

- What are the candidate plans?
  - Choose an arbitrary each leaf node of the join tree
  - Augment each node with output attributes in its subtree
  - We will get a free-connex tree decomposition



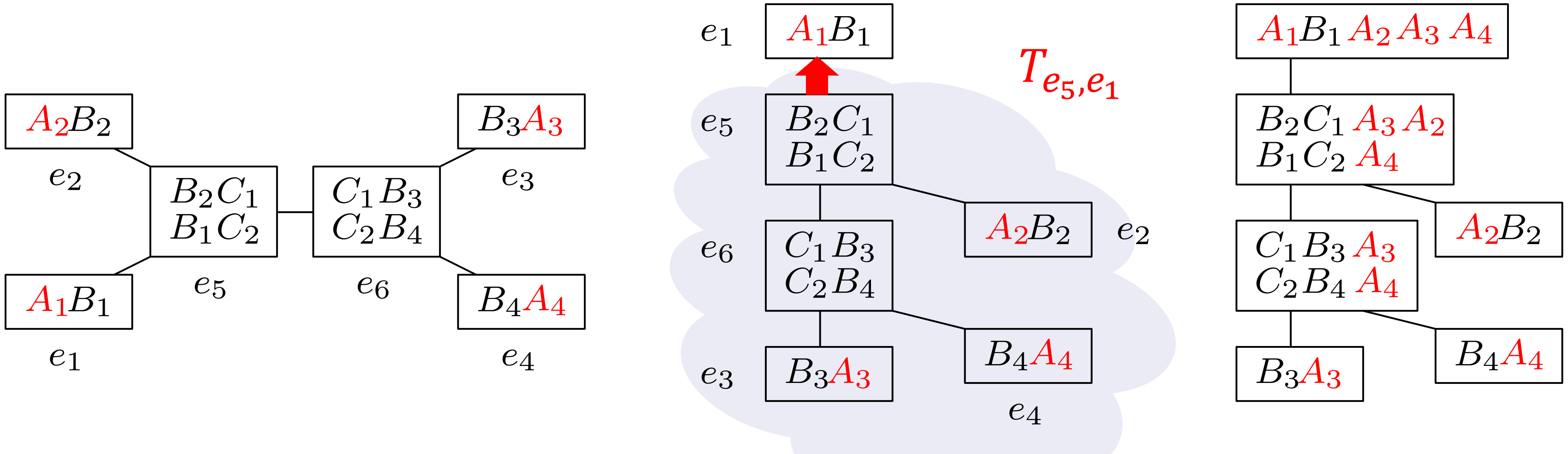
# Minimal Sub-instance in the Partition

- What is the minimal sub-instance and which query plan to choose?




- If every $b \in B_1$ can be joined with at most $OUT^{\frac{3}{4}}$ distinct tuples over $(A_2, A_3, A_4)$

# Edge Labeling



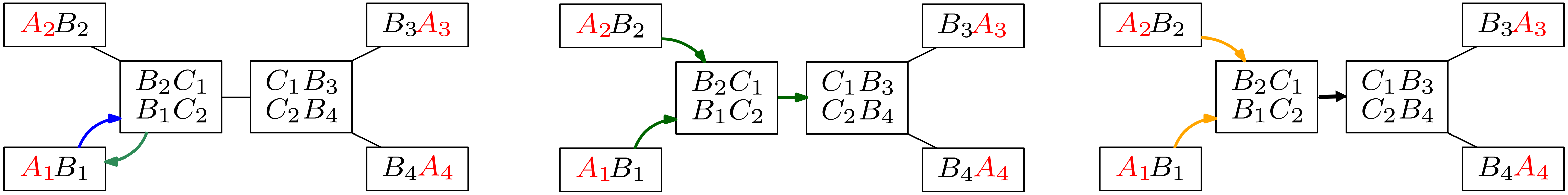

“budget” $\phi_{e_5,e_1} = \dfrac{|\#\text{ leaves in } T_{e_5,e_1}|}{\text{fn−fhtw}} = \dfrac{3}{4}$

- Edge $(e_5, e_1)$ is large if every $b \in B_1$ joins $\geq OUT^{\phi_{e_5,e_1}}$ tuples over $(A_2, A_3, A_4)$
- Edge $(e_5, e_1)$ is small if every $b \in B_1$ joins $< OUT^{\phi_{e_5,e_1}}$ tuples over $(A_2, A_3, A_4)$
  - Edge $(e_5, e_1)$ is limited if there are $\leq OUT^{\phi_{e_5,e_1}}$ tuples over $(A_2, A_3, A_4)$

# Partition – Label Edges

- It is expensive to compute the labels straightforwardly
- Inference rules:

unknown?

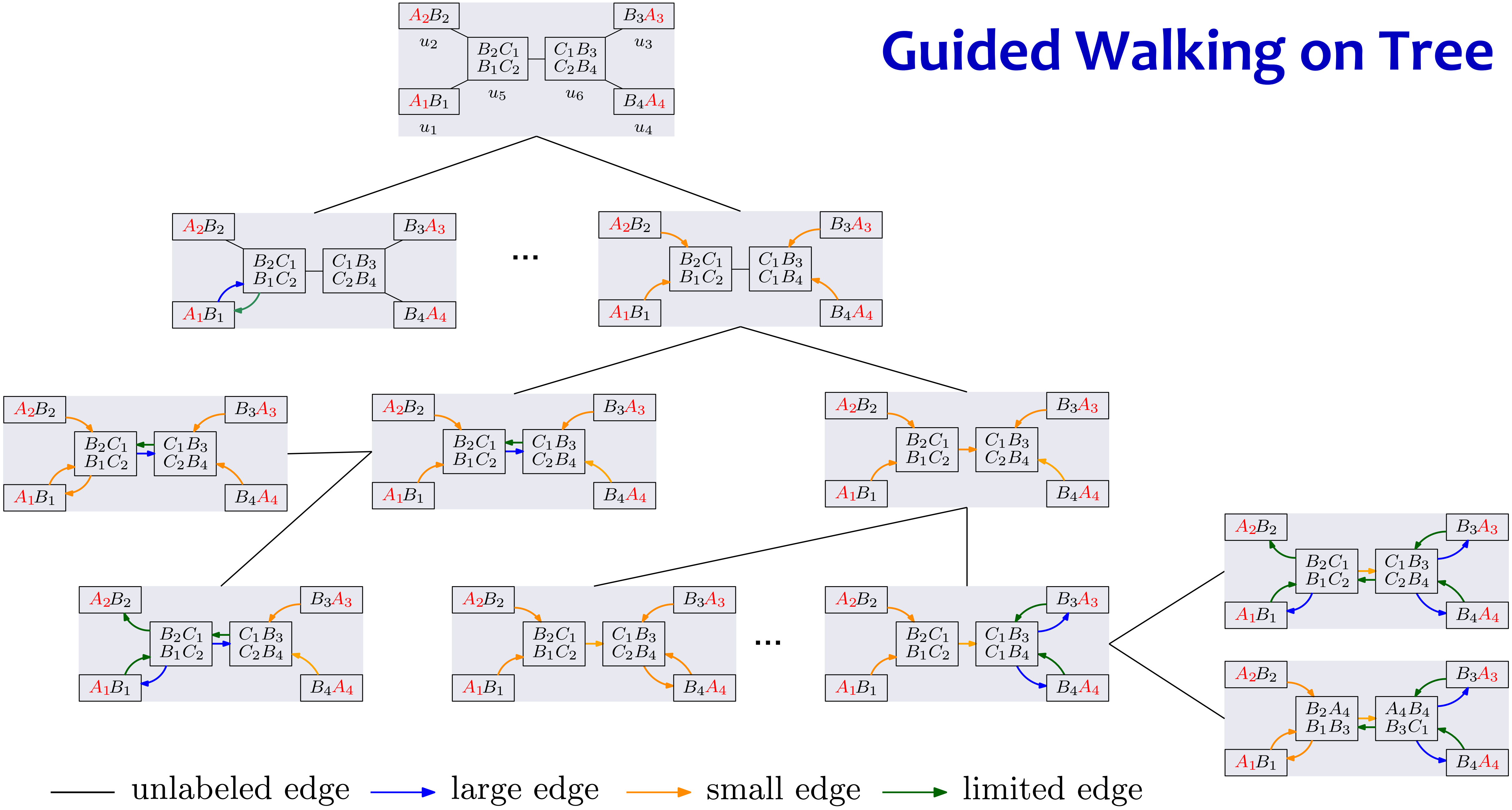



Rule 1 - Large reverse Limited

Rule 2 - Limited imply Limited

But we can partition!

# Guided Walking on Tree



# Summary

- An output-optimal algorithm for computing acyclic join-aggregate queries
  - The power of hybrid strategies
- Free-connex fractional hypertree width
- It remains open how to implement this algorithm in practice!