# Multi-join Query Evaluation on Big Data
# Lecture 3

Dan Suciu

March, 2015

# Multi-join Query Evaluation – Outline

Part 1 Optimal Sequential Algorithms.

Part 2 Lower bounds for Parallel Algorithms.

Part 3 Optimal Parallel Algorithms.

Part 3 Data Skew.

# Summary so far

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad\quad$ $|R_1| = m_1, \ldots, |R_\ell| = m_\ell$

**Sequential World** Cost: output size of $Q$

- Upper bound $AGM(Q) = m^{\rho^*}$. Fractional edge cover.
- Lower bound (tightness): fractional vertex packing
- Generic-join algorithm.

**Parallel World** Cost: communication.

- 1-round, skew-free, equal-cardinalities.
- Lower bound $m/p^{1/\tau^*}$. Fractional edge packing
- Upper bound: fractional vertex cover.
- HyperCube algorithm.

# Summary so far

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad\quad$ $|R_1| = m_1, \ldots, |R_\ell| = m_\ell$

**Sequential World** Cost: output size of $Q$

- Upper bound $AGM(Q) = m^{\rho^*}$. Fractional edge cover.
- Lower bound (tightness): fractional vertex packing
- Generic-join algorithm.

**Parallel World** Cost: communication.

- 1-round, skew-free, equal-cardinalities.
- Lower bound $m/p^{1/\tau^*}$. Fractional edge packing
- Upper bound: fractional vertex cover.
- HyperCube algorithm.

# Summary so far

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad\quad$ $|R_1| = m_1, \ldots, |R_\ell| = m_\ell$

**Sequential World** Cost: output size of $Q$

- Upper bound $AGM(Q) = m^{\rho^*}$. Fractional edge cover.
- Lower bound (tightness): fractional vertex packing
- Generic-join algorithm.

**Parallel World** Cost: communication.

- 1-round, skew-free, equal-cardinalities.
- Lower bound $m/p^{1/\tau^*}$. Fractional edge packing
- Upper bound: fractional vertex cover.
- HyperCube algorithm.

# Outline of Lecture 3

- HyperCube Algorithm for arbitrary cardinalities
- Lower bound formula for arbitrary cardinalities
- Prove that they are equal
- Summary

Will consider only databases without skew

# Why Databases without Skew Matter

- In practice, skewed values are detected and treated separately; cost should be a function of the degree of skew.
- Example: join $Q(x, y, z) = R(x, y), S(y, z)$.
  Without skew: $L = m/p$. (Common case)
  With skew, as bad as cartesian product: $L \geq m/p^{1/2}$.
- In general, for any query $Q$:
  Without skew: $L = m/p^{1/\tau^*}$.
  With skew: $L \geq m/p^{1/\rho^*}$ (lecture 2)

# Review of the HyperCube Algorithm

- Afrati and Ullman described in EDBT'2010 an algorithm for computing any conjunctive in one MapReduce job. Same as a one-round algorithm on the MPC model. Later, it was called the *Shares* algorithm.

- Beame, Koutris, and S. analyzed in PODS'2013 and PODS '2014 the parameters for the algorithm, and called it the *HyperCube* algorithm. We will use this name.

# Review of the HyperCube Algorithm

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Compute $Q$ on $p$ servers.

# Review of the HyperCube Algorithm

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad$ $|R_1| = m_1, \ldots, |R_\ell| = m_\ell$
Compute $Q$ on $p$ servers.

Organize the $p$ servers in a hypercube: $[p] = [p_1] \times \cdots \times [p_k]$.
The numbers $p_1, \ldots, p_k$ are called *shares*.
Choose $k$ independent hash functions $h_1, \ldots, h_k$

Round 1 Each server sends each tuple $R_j(x_{j_1}, x_{j_2}, \ldots)$ to all servers whose coordinates $j_1, j_2, \ldots$ are $h_{j_1}(x_{j_1}), h_{j_2}(x_{j_2}), \ldots$ and broadcasts along the missing dimensions. Then, each server computes $Q$ on its local data.

Problem: compute the shares $p_1, \ldots, p_k$.

# They HyperCube Algorithm – Computing the Shares

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

## The Shares-Problem

Find shares $p_1, \ldots, p_k$ s.t. $\prod_i p_i = p$ and the load is minimized.

Number of tuples that a server receives from $R_j$ is: $\frac{m_j}{\prod_{i \in R_j} p_i}$

# They HyperCube Algorithm – Computing the Shares

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

## The Shares-Problem

Find shares $p_1, \ldots, p_k$ s.t. $\prod_i p_i = p$ and the load is minimized.

Number of tuples that a server receives from $R_j$ is: $\frac{m_j}{\prod_{i \in R_j} p_i}$

[Afrati&Ullman'10] Optimize $L = \sum_j \frac{m_j}{\prod_{i \in R_j} p_i}$. Non-linear.

# They HyperCube Algorithm – Computing the Shares

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

## The Shares-Problem

Find shares $p_1, \ldots, p_k$ s.t. $\prod_i p_i = p$ and the load is minimized.

Number of tuples that a server receives from $R_j$ is: $\frac{m_j}{\prod_{i \in R_j} p_i}$

[Afrati&Ullman'10] Optimize $L = \sum_j \frac{m_j}{\prod_{i \in R_j} p_i}$. Non-linear.

[Beame'14] Optimize $L = \max_j \frac{m_j}{\prod_{i \in R_j} p_i}$:

The Shares Problem:

$$\begin{array}{ll} & \text{minimize } L \\ & p_1 \cdot p_2 \cdots p_k \leq p \\ \forall j: & L \geq \frac{m_j}{\prod_{i \in R_j} p_i} \end{array}$$

Will show that this is equivalent to a linear optimization problem.

# E-Shares: A Linear Optimization Problem

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Optimization problem: find shares $p_1, \ldots, p_\ell$ such that

| | The Shares Problem | The E-Shares Linear Problem |
|---|---|---|
| Parameter:<br>Shares<br>Sizes<br>Load | Value:<br>$p_1, \ldots, p_k$<br>$m_1, \ldots, m_\ell$<br>$L$ | |
| Optimize: | minimize $L$<br>$p_1 \cdot p_2 \cdots p_k \leq p$<br>$\forall j: \quad L \geq \frac{m_j}{\prod_{i \in R_j} p_i}$ | |

# E-Shares: A Linear Optimization Problem

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Optimization problem: find shares $p_1, \ldots, p_\ell$ such that

| | The Shares Problem | The E-Shares Linear Problem |
|---|---|---|
| Parameter:<br>Shares<br>Sizes<br>Load | Value:<br>$p_1, \ldots, p_k$<br>$m_1, \ldots, m_\ell$<br>$L$ | $\log_p$ Value: |
| Optimize: | minimize $L$<br>$p_1 \cdot p_2 \cdots p_k \leq p$<br>$\forall j: \quad L \geq \frac{m_j}{\prod_{i \in R_j} p_i}$ | |

# E-Shares: A Linear Optimization Problem

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Optimization problem: find shares $p_1, \ldots, p_\ell$ such that

| | The Shares Problem | The E-Shares Linear Problem |
|---|---|---|
| Parameter:<br>Shares<br>Sizes<br>Load | Value:<br>$p_1, \ldots, p_k$<br>$m_1, \ldots, m_\ell$<br>$L$ | $\log_p$ Value:<br>$e_1, \ldots, e_k$<br>$\mu_1, \ldots, \mu_\ell$<br>$\lambda$ |
| Optimize: | minimize $L$<br>$p_1 \cdot p_2 \cdots p_k \leq p$<br>$\forall j: \quad L \geq \frac{m_j}{\prod_{i \in R_j} p_i}$ | |

# E-Shares: A Linear Optimization Problem

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Optimization problem: find shares $p_1, \ldots, p_\ell$ such that

| | The Shares Problem | The E-Shares Linear Problem |
|---|---|---|
| Parameter:<br>Shares<br>Sizes<br>Load | Value:<br>$p_1, \ldots, p_k$<br>$m_1, \ldots, m_\ell$<br>$L$ | $\log_p$ Value:<br>$e_1, \ldots, e_k$<br>$\mu_1, \ldots, \mu_\ell$<br>$\lambda$ |
| Optimize: | minimize $L$<br>$p_1 \cdot p_2 \cdots p_k \leq p$<br>$\forall j: \quad L \geq \frac{m_j}{\prod_{i \in R_j} p_i}$ | minimize $\lambda$<br>$-e_1 - e_2 - \ldots - e_k \geq -1$<br>$\forall j: \quad \lambda + \sum_{i: i \in R_j} e_i \geq \mu_j$ |

# E-Shares: A Linear Optimization Problem

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad |R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Optimization problem: find shares $p_1, \ldots, p_\ell$ such that

| | The Shares Problem | The E-Shares Linear Problem |
|---|---|---|
| Parameter:<br>Shares<br>Sizes<br>Load | Value:<br>$p_1, \ldots, p_k$<br>$m_1, \ldots, m_\ell$<br>$L$ | $\log_p$ Value:<br>$e_1, \ldots, e_k$<br>$\mu_1, \ldots, \mu_\ell$<br>$\lambda$ |
| Optimize: | minimize $L$<br>$p_1 \cdot p_2 \cdots p_k \leq p$<br>$\forall j: \quad L \geq \frac{m_j}{\prod_{i \in R_j} p_i}$ | minimize $\lambda$<br>$-e_1 - e_2 - \ldots - e_k \geq -1$<br>$\forall j: \quad \lambda + \sum_{i: i \in R_j} e_i \geq \mu_j$ |

Optimal shares: $p_1 = p^{e_1^*}, \ldots, p_k = p^{e_k^*}$ $\quad$ Optimal load: $L = p^{\lambda^*}$

# Discussion

- For equal-cardinalities, $L = m/p^{1/\tau^*}$. Speedup given by the optimal fractional edge packing.
- What is the speedup now? The E-Shares formula $L = p^{\lambda^*}$ is not insightful, as $\lambda^*$ depends on $\mu_1, \ldots, \mu_\ell$.
- Goal: analyze how $L$ depends on $p$ (speedup) and on the cardinalities $m_1, \ldots, m_\ell$.

# Review: Fractional Vertex Cover / Edge Packing

Hypergraph: $Q = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ Nodes: $x_1, \ldots, x_k$, edges: $R_1, \ldots, R_\ell$.
*Vertex cover*= set of nodes such that every edge contains a node in the set.

## Definition

A *fractional vertex cover* of $Q$ is a sequence $v_1 \geq 0, \ldots, v_k \geq 0$ such that:

$$\forall j : \sum_{i : x_i \in R_j} v_i \geq 1$$

*Edge packing*= set of edges with no vertex in common.

## Definition

A *fractional edge packing* of $Q$ is a sequence $u_1 \geq 0, \ldots, u_\ell \geq 0$ such that:

$$\forall i : \sum_{j : x_i \in R_j} u_j \leq 1$$

By duality: $\min_{\mathbf{v}} \sum_i v_i = \max_{\mathbf{u}} \sum_j u_j = \tau^*$

# Review: Fractional Vertex Cover / Edge Packing

Hypergraph: $Q = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ Nodes: $x_1, \ldots, x_k$, edges: $R_1, \ldots, R_\ell$.
*Vertex cover*= set of nodes such that every edge contains a node in the set.

## Definition

A *fractional vertex cover* of $Q$ is a sequence $v_1 \geq 0, \ldots, v_k \geq 0$ such that:

$$\forall j : \sum_{i : x_i \in R_j} v_i \geq 1$$

*Edge packing*= set of edges with no vertex in common.

## Definition

A *fractional edge packing* of $Q$ is a sequence $u_1 \geq 0, \ldots, u_\ell \geq 0$ such that:

$$\forall i : \sum_{j : x_i \in R_j} u_j \leq 1$$

By duality: $\min_{\mathbf{v}} \sum_i v_i = \max_{\mathbf{u}} \sum_j u_j = \tau^*$

# Adding $v_0$

We add a variable $v_0$ to the fractional vertex cover, which becomes the new objective:

| Fractional Vertex Cover | Fractional Vertex Cover with $v_0$ |
|---|---|
| $\begin{array}{ll} & \text{minimize } \sum_i v_i \\ \forall j: & \sum_{i:x_i \in R_j} v_i \geq 1 \end{array}$ | $\begin{array}{ll} & \text{minimize } v_0 \\ & -\sum_i v_i \geq -v_0 \\ \forall j: & \sum_{i:x_i \in R_j} v_i \geq 1 \end{array}$ |

# Checking the Equal-Cardinalities Case

When $m_1 = \ldots = m_k = m$, we know $L = m/p^{1/\tau^*}$. What does E-Shares do?

| E-Shares LP | Fractional Vertex Cover with $v_0$ |
|---|---|
| minimize $\lambda$ | minimize $v_0$ |
| $-\sum_i e_i \geq -1$ | $-\sum_i v_i \geq -v_0$ |
| $\forall j: \quad \sum_{i:i\in R_j} e_i \geq \mu - \lambda$ | $\forall j: \quad \sum_{i:i\in R_j} v_i \geq 1$ |

The following are 1-1 inverse mappings between feasible solutions:

$$(\lambda, e_1, \ldots, e_k) \quad \mapsto \quad \left(v_0 = \frac{1}{\mu-\lambda}, v_1 = \frac{e_1}{\mu-\lambda}, \ldots, v_k = \frac{e_k}{\mu-\lambda}\right)$$

$$\left(\lambda = \mu - \frac{1}{v_0}, e_1 = \frac{v_1}{v_0}, \ldots, e_k = \frac{v_k}{v_0}\right) \quad \mapsfrom \quad (v_0, v_1, \ldots, v_k)$$

The optimal load of E-Shares is $L = p^{\lambda^*} = \frac{p^\mu}{p^{1/v_0^*}} = m/p^{1/\tau^*}$, same as before.

# Goal: Analyze $L$ for Arbitrary Cardinalities

- To analyze the load $L$ for arbitrary cardinalities $m_1, \ldots, m_\ell$, we first give a lower bound formula. This is a closed formula, based on fractional edge packings.
- As for the AGM bound, I will first give a simple intuition behind the lower bound, based on cartesian products.
- Next steps: (1) prove that the formula is a lower bound for any algorithm $A$: this generalizes the proof from Lecture 2 and is omitted; (2) prove that the formula is equal to E-Shares: we will use duality.

# Optimal Load $L$ for Cartesian Products

Problem: find the optimal load for a cartesian product:

$$Q(x, y) = R(x), S(y), \quad |R| = m_1, |S| = m_2$$

Solution: organize the servers in a matrix $[p] = [p_1] \times [p_2]$
Send $R(x) \to (h_1(x), *)$, send $S(y) \to (*, h_2(y))$.



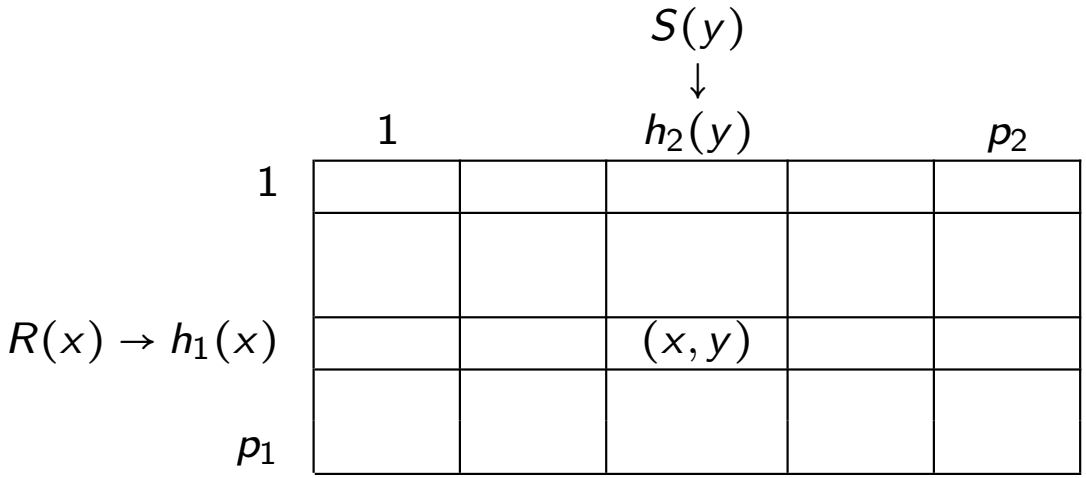

Load: $L = \frac{m_1}{p_1} + \frac{m_2}{p_2} \geq 2\sqrt{\frac{m_1 m_2}{p}}$ = optimal.

# Optimal Load $L$ for Cartesian Products

Problem: find the optimal load for a cartesian product:

$$Q(x, y) = R(x), S(y), \quad |R| = m_1, |S| = m_2$$

Solution: organize the servers in a matrix $[p] = [p_1] \times [p_2]$
Send $R(x) \to (h_1(x), *)$, send $S(y) \to (*, h_2(y))$.



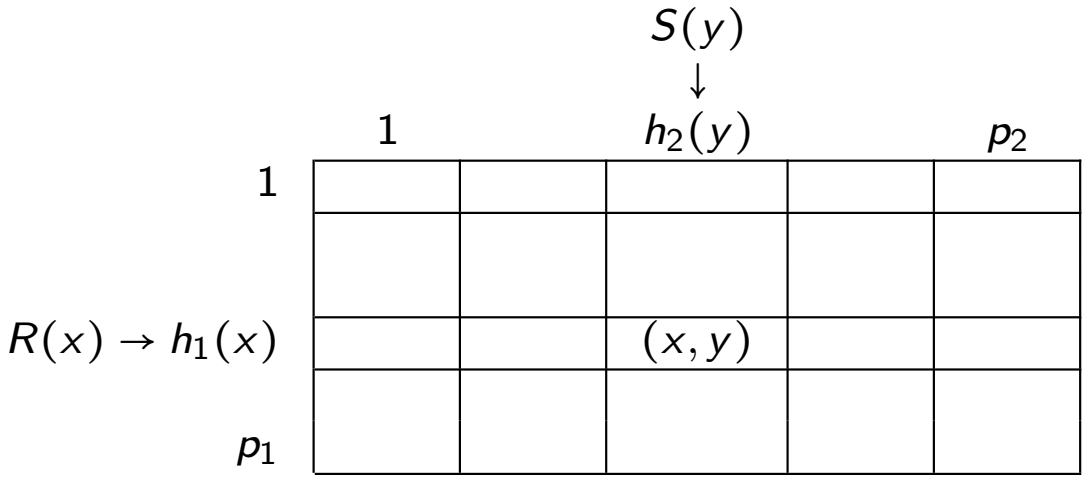

Load: $L = \frac{m_1}{p_1} + \frac{m_2}{p_2} \geq 2\sqrt{\frac{m_1 m_2}{p}}$ = optimal.

# Optimal Load $L$ for Cartesian Products

Problem: find the optimal load for a cartesian product:

$$Q(x, y) = R(x), S(y), \quad |R| = m_1, |S| = m_2$$

Solution: organize the servers in a matrix $[p] = [p_1] \times [p_2]$
Send $R(x) \to (h_1(x), *)$, send $S(y) \to (*, h_2(y))$.




Load: $L = \frac{m_1}{p_1} + \frac{m_2}{p_2} \geq 2\sqrt{\frac{m_1 m_2}{p}}$ = optimal.

# Side Note: Criticism of the MapReduce Model

- Load of cartesian product $m/p^{1/2}$. Data is replicated $p^{1/2}$ times.
- Ullman [ACM Crossroads'12] described the *drug interaction problem*, using MapReduce: Compute a cartesian product, then apply a UDF to all pairs.
- In MapReduce, $p$ corresponds to the *number of reducers*, which the programmer can set at will. Recommendation: use as many reducers as possible.
- "Obvious" solution: $p = m^2$ reducers, meaning the entire data is replicated $m$ times! Failed dramatically.
- Lesson: the MapReduce model hides the true number of severs $p$, which makes it difficult to design and analyze algorithms.

# Optimal Load $L$ for Cartesian Products

## Proposition

*Any algorithm computing $R \times S$ has load $\geq \sqrt{m_1 m_2 / p}$.*

## Proof.

Let $L$ be the load.
Any single server can report $\leq L^2$ pairs $(x, y)$.
The $p$ servers can report $\leq pL^2$ pairs
Hence, $pL^2 \geq m_1 m_2$. □

## Proposition

*Any algorithm for computing $R_1 \times R_2 \times \cdots \times R_k$ has load $\geq k \left( \frac{m_1 m_2 \cdots m_k}{p} \right)^{1/k}$*

# Optimal Load $L$ for Cartesian Products

Let $Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$.

## Theorem

*Let $R_{j_1}, \ldots, R_{j_k}$ be an edge packing. Any algorithm for $Q$ has load*

$$L \geq k \left( \frac{m_{j_1} m_{j_2} \cdots m_{j_k}}{p} \right)^{1/k}$$

## Proof.

Any algorithm that computes $Q$ correctly, must send every tuple in $R_{j_1} \times \ldots \times R_{j_k}$ to some server. Suppose not: no server receives a tuple $t = (\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_k})$. Modify the other relations by adding new tuples, to make $t$ part of a query answer: but the algorithm fails to find it. □

# The Load of the HyperCube Algorithm

Denote

$$L(\mathbf{u}) = \left( \frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p} \right)^{\frac{1}{u_0}} \quad \text{where } u_0 = u_1 + \ldots + u_\ell \quad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u})$$

## Theorem (Beame'14)

1. *Let* $\mathbf{u}$ *be any fractional edge packing. Any algorithm computing* $Q$ *has a load:* $\Omega(L(\mathbf{u}))$*, even on databases of partial matchings*[a]*.*
2. *The optimal load of HyperCube algorithm is* $L_{lower}$*. Hence, optimal.*

[a] Subset $R_j \subseteq [n]^r$ of size $m_j$ that is a matching up to renaming of values.

The proof of (1) generalizes our previous proof for matchings (omitted).
We will prove (2) today, but first, let's discuss the formula.
Problem: prove that $\texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$ is a vertex of the edge packing polytope.

# The Load of the HyperCube Algorithm

Denote

$$L(\mathbf{u}) = \left( \frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p} \right)^{\frac{1}{u_0}} \quad \text{where } u_0 = u_1 + \ldots + u_\ell \quad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u})$$

## Theorem (Beame'14)

1. *Let* $\mathbf{u}$ *be any fractional edge packing. Any algorithm computing* $Q$ *has a load:* $\Omega(L(\mathbf{u}))$*, even on databases of partial matchings*[a]*.*
2. *The optimal load of HyperCube algorithm is* $L_{lower}$*. Hence, optimal.*

[a] Subset $R_j \subseteq [n]^r$ of size $m_j$ that is a matching up to renaming of values.

The proof of (1) generalizes our previous proof for matchings (omitted).
We will prove (2) today, but first, let's discuss the formula.
Problem: prove that $\texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$ is a vertex of the edge packing polytope.

# Example

$Q(x, y, z) = R(x, y), S(y, z), T(z, x)$ $L(\mathbf{u}) = (m_1^{u_1} m_2^{u_2} m_3^{u_3}/p)^{1/(u_1+u_2+u_3)}$:

1

# Example

$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$ $\quad L(\mathbf{u}) = (m_1^{u_1} m_2^{u_2} m_3^{u_3}/p)^{1/(u_1+u_2+u_3)}$:

| $\mathbf{u}$ | $L(\mathbf{u})$ |
|---|---|
| $(1/2, 1/2, 1/2)$ | $(m_1 m_2 m_3)^{1/3}/p^{2/3}$ |
| | |
| | |
| | |
| | |

$L = \max_{\mathbf{u}} L_{\text{lower}}(\mathbf{u})$

1

# Example

$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$ $\quad L(\mathbf{u}) = (m_1^{u_1} m_2^{u_2} m_3^{u_3}/p)^{1/(u_1+u_2+u_3)}$:

| $\mathbf{u}$ | $L(\mathbf{u})$ |
|---|---|
| $(1/2, 1/2, 1/2)$ | $(m_1 m_2 m_3)^{1/3}/p^{2/3}$ |
| $(1, 0, 0)$ | $m_1/p$ |
| | |
| | |
| | |

$L = \max_{\mathbf{u}} L_{\text{lower}}(\mathbf{u})$

1

# Example

$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$ $\quad L(\mathbf{u}) = (m_1^{u_1} m_2^{u_2} m_3^{u_3}/p)^{1/(u_1+u_2+u_3)}$:

| $\mathbf{u}$ | $L(\mathbf{u})$ |
|---|---|
| $(1/2, 1/2, 1/2)$ | $(m_1 m_2 m_3)^{1/3}/p^{2/3}$ |
| $(1,0,0)$ | $m_1/p$ |
| $(0,1,0)$ | $m_2/p$ |
| $(0,0,1)$ | $m_3/p$ |
| $(0,0,0)$ | 0 (why[1]?) |

$L = \max_{\mathbf{u}} L_{\text{lower}}(\mathbf{u})$

[1] $L(\mathbf{u}) \leq \max(m_1, m_2, m_3)/p^{1/(u_1+u_2+u_3)} \to 0$ when $u_1 + u_2 + u_3 \to 0$.

# Example

$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$ $\quad L(\mathbf{u}) = (m_1^{u_1} m_2^{u_2} m_3^{u_3}/p)^{1/(u_1+u_2+u_3)}$:

| $\mathbf{u}$ | $L(\mathbf{u})$ |
|---|---|
| $(1/2, 1/2, 1/2)$ | $(m_1 m_2 m_3)^{1/3}/p^{2/3}$ |
| $(1,0,0)$ | $m_1/p$ |
| $(0,1,0)$ | $m_2/p$ |
| $(0,0,1)$ | $m_3/p$ |
| $(0,0,0)$ | 0 (why[1]?) |

$L = \max_{\mathbf{u}} L_{\text{lower}}(\mathbf{u})$

Suppose $m_1 < m_2 = m_3$, then $L = \max(m_3/p, (m_1 m_2 m_3)^{1/3}/p^{2/3})$:

$p \leq m_3/m_1$ Then $L_{\text{lower}} = m_3/p$, linear speedup.
HyperCube: compute $S \bowtie T$, broadcast $R$.

$p > m_3/m_1$ Then $L_{\text{lower}} = (m_1 m_2 m_3)^{1/3}/p^{2/3}$, speedup decreased to $1/p^{2/3}$.

[1] $L(\mathbf{u}) \leq \max(m_1, m_2, m_3)/p^{1/(u_1+u_2+u_3)} \to 0$ when $u_1 + u_2 + u_3 \to 0$.

# Properties of Hypercube – Informal

- HyperCube takes advantage of un-equal cardinalities $m_1, \ldots, m_\ell$: allocate fewer shares to the smalle realtions, or even broadcast them.

- When $p$ increases, the utility of the smaller relations diminishes: the speedup decreases, eventually reaching $1/p^{\tau^*}$.

# Properties of Hypercube – Formal

$$L(\mathbf{u}) = \left(\frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p}\right)^{\frac{1}{u_0}} \qquad u_0 = \sum_i u_i \qquad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u}) \qquad \mathbf{u}^* = \texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$$

- Speedup is $1/p^{1/u_0^*}$.

# Properties of Hypercube – Formal

$$L(\mathbf{u}) = \left(\frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p}\right)^{\frac{1}{u_0}} \qquad u_0 = \sum_i u_i \qquad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u}) \qquad \mathbf{u}^* = \texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$$

- Speedup is $1/p^{1/u_0^*}$.
- As $p$ increases, $u_0^*$ increases too.

  (Proof: if $u_0 < u_0'$, then $L(\mathbf{u}') - L(\mathbf{u}) = a/p^{1/u_0'} - b/p^{1/u_0}$ strictly increasing in $p$.)

  Meaning: speedup deteriorates, eventually reaching $1/p^{1/\tau^*}$.

# Properties of Hypercube – Formal

$$L(\mathbf{u}) = \left(\frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p}\right)^{\frac{1}{u_0}} \qquad u_0 = \sum_i u_i \qquad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u}) \qquad \mathbf{u}^* = \texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$$

- Speedup is $1/p^{1/u_0^*}$.
- As $p$ increases, $u_0^*$ increases too.

  (Proof: if $u_0 < u_0'$, then $L(\mathbf{u}') - L(\mathbf{u}) = a/p^{1/u_0'} - b/p^{1/u_0}$ strictly increasing in $p$.)

  Meaning: speedup deteriorates, eventually reaching $1/p^{1/\tau^*}$.
- If $m_j < L_{\text{lower}}$, then $u_j^* = 0$.

  (Proof. $L(u_j) = \exp((au_j + b)/(cu_j + d))$, with $a, b, c, d > 0$;
  Since $L_{\text{lower}} = L(u_j^*) > \lim_{u_j \to \infty} L(u_j) = m_j$, it is strictly decreasing on $(0, \infty)$.)

  Meaning: relations smaller than the optimal load are broadcast.

# Properties of Hypercube – Formal

$$L(\mathbf{u}) = \left(\frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p}\right)^{\frac{1}{u_0}} \qquad u_0 = \sum_i u_i \qquad L_{\text{lower}} = \max_{\mathbf{u}} L(\mathbf{u}) \qquad \mathbf{u}^* = \texttt{argmax}_{\mathbf{u}} L(\mathbf{u})$$

- Speedup is $1/p^{1/u_0^*}$.
- As $p$ increases, $u_0^*$ increases too.

  (Proof: if $u_0 < u_0'$, then $L(\mathbf{u}') - L(\mathbf{u}) = a/p^{1/u_0'} - b/p^{1/u_0}$ strictly increasing in $p$.)

  Meaning: speedup deteriorates, eventually reaching $1/p^{1/\tau^*}$.
- If $m_j < L_{\text{lower}}$, then $u_j^* = 0$.

  (Proof. $L(u_j) = \exp((au_j + b)/(cu_j + d))$, with $a, b, c, d > 0$;
  Since $L_{\text{lower}} = L(u_j^*) > \lim_{u_j \to \infty} L(u_j) = m_j$, it is strictly decreasing on $(0, \infty)$.)

  Meaning: relations smaller than the optimal load are broadcast.
- Let $m_{j_0} = \max_j m_j$. If $m_j < m_{j_0}/p$, then $u_j^* = 0$ (since $m_{j_0}/p \leq L_{\text{lower}}$).
  Meaning: relations less than $1/p$ of some other relation are broadcast.

# Proof of Equivalence

$Q(\mathbf{x}) = R_1(\mathbf{x}_1), \ldots, R_\ell(\mathbf{x}_\ell)$ $\quad$ $|R_1| = m_1, \ldots, |R_\ell| = m_\ell$

Recall:

E-Shares:

$$\begin{aligned} & \text{minimize } \lambda \\ & -\textstyle\sum_i e_i \geq -1 \\ \forall j: \quad & \lambda + \textstyle\sum_{i:i\in R_j} e_i \geq \mu_j \end{aligned}$$

Where $\mu_1 = \log_p m_1, \ldots, \mu_\ell = \log_p \mu_\ell$.

The shares and load of HyperCube: $p_1 = p^{e_1^*}, \ldots, p_k = p^{e_k^*}$ $\quad$ $L = p^{\lambda^*}$

# Proof of Equivalence

$$L_{\text{lower}} = \max_{\mathbf{u}} \left( \frac{m_1^{u_1} \cdots m_\ell^{u_\ell}}{p} \right)^{\frac{1}{u_0}} \qquad u_0 = \sum_j u_j$$

Apply $\log_p$, and change to $\sum_j u_j \leq u_0$ w.l.o.g. (why?):

Log-$L_{\text{lower}}$:

$$\begin{aligned} & \text{maximize } (\mu_1 u_1 + \ldots + \mu_\ell u_\ell - 1)/u_0 \\ & u_1 + \ldots + u_\ell \leq u_0 \\ \forall i: \quad & \textstyle\sum_{j: i \in R_j} u_j \leq 1 \end{aligned}$$

Where $\mu_1 = \log_p m_1, \ldots, \mu_\ell = \log_p \mu_\ell$.
This is *not* a linear program.

# Proof of Equivalence

| E-Shares | Dual E-Shares | Log-$L_{\text{lower}}$ |
|---|---|---|
| $\min \lambda$<br>$-\sum_i e_i \geq -1$<br>$\forall j: \ \lambda + \sum_{i:i \in R_j} e_i \geq \mu_j$ | $\max \mu_1 f_1 + \ldots + \mu_\ell f_\ell - f_0$<br>$f_1 + \ldots + f_\ell \leq 1$<br>$\forall i: \ \sum_{j:i \in R_j} f_j \leq f_0$ | $\max (\mu_1 u_1 + \ldots + \mu_\ell u_\ell - 1)/u_0$<br>$u_1 + \ldots + u_\ell \leq u_0$<br>$\forall i: \ \sum_{j:i \in R_j} u_j \leq 1$ |

E-Shares and Dual E-Shares are duals: optimal loads are equal.

Dual E-Shares and Log-$L_{\text{lower}}$ have the same optimal load, by the following mapping between feasible solutions:

$$(f_0, f_1, \ldots, f_\ell) \quad \mapsto \quad (u_0 = 1/f_0, u_1 = f_1/f_0, \ldots, u_\ell = f_\ell/f_0)$$
$$(f_0 = 1/u_0, f_1 = u_1/u_0, \ldots, f_\ell = u_\ell/u_0) \quad \mapsfrom \quad (u_1, u_2, \ldots, u_\ell, u_0)$$

# Summary of Lecture 3

- The roles of Primal/Dual:
  - The primal LP describes the HyperCube parallel algorithm; Fractional Vertex Cover.
  - The dual LP describes the Lower Bound for Parallel Evaluation; Fractional Edge Packing.
- AGM bound versus Lower Bound for parallel evaluation:
  - Vertex packing / Edge cover $\rho^*$ versus Vertex cover / Edge packing $\tau^*$.
  - General (skewed) databases versus skew-free databases.
  - Both formulas generalize simple observations for cartesian products.
- Skewed data: will discuss in Lecture 4 (mostly open problems)
- Multi rounds: will discuss briefly in lecture 4 (almost entirely open).

# Summary of Lecture 3

- The roles of Primal/Dual:
  - The primal LP describes the HyperCube parallel algorithm; Fractional Vertex Cover.
  - The dual LP describes the Lower Bound for Parallel Evaluation; Fractional Edge Packing.
- AGM bound versus Lower Bound for parallel evaluation:
  - Vertex packing / Edge cover $\rho^*$ versus Vertex cover / Edge packing $\tau^*$.
  - General (skewed) databases versus skew-free databases.
  - Both formulas generalize simple observations for cartesian products.
- Skewed data: will discuss in Lecture 4 (mostly open problems)
- Multi rounds: will discuss briefly in lecture 4 (almost entirely open).

# Summary of Lecture 3

- The roles of Primal/Dual:
  - The primal LP describes the HyperCube parallel algorithm; Fractional Vertex Cover.
  - The dual LP describes the Lower Bound for Parallel Evaluation; Fractional Edge Packing.
- AGM bound versus Lower Bound for parallel evaluation:
  - Vertex packing / Edge cover $\rho^*$ versus Vertex cover / Edge packing $\tau^*$.
  - General (skewed) databases versus skew-free databases.
  - Both formulas generalize simple observations for cartesian products.
- Skewed data: will discuss in Lecture 4 (mostly open problems)
- Multi rounds: will discuss briefly in lecture 4 (almost entirely open).

# Summary of Lecture 3

- The roles of Primal/Dual:
  - The primal LP describes the HyperCube parallel algorithm; Fractional Vertex Cover.
  - The dual LP describes the Lower Bound for Parallel Evaluation; Fractional Edge Packing.
- AGM bound versus Lower Bound for parallel evaluation:
  - Vertex packing / Edge cover $\rho^*$ versus Vertex cover / Edge packing $\tau^*$.
  - General (skewed) databases versus skew-free databases.
  - Both formulas generalize simple observations for cartesian products.
- Skewed data: will discuss in Lecture 4 (mostly open problems)
- Multi rounds: will discuss briefly in lecture 4 (almost entirely open).