Review: *Skew Strikes Back: New Developments in the Theory of Join Algorithms*

Minsi Lu, 21168814

1 Summary and Problem Statement

This paper is a theory-focused survey that addresses a central question in database query processing: **Can we design join algorithms that are *worst-case optimal*, i.e., whose runtime matches the tightest possible bound on output size for *all* inputs?**

Traditional database systems execute multiway joins as a sequence of binary joins, often suffering severe performance degradation in the presence of skewed data or cyclic query structures. The Atserias–Grohe–Marx (AGM) bound provides a tight upper bound on the output size of a join query, but most classical join plans fail to achieve this bound in the worst case.

The authors review recent developments that:

1. Establish non-trivial bounds (e.g., $N^{1.5}$ for triangle queries) for all join queries.
2. Present algorithms that achieve these bounds, termed *worst-case optimal join algorithms*, such as NPRR and the Leapfrog Triejoin algorithm.
3. Provide a simpler, unified description of these algorithms at the theoretical level via the *Generic-Join* algorithm.

2 Key Insights

2.1 Start from Triangle Query: Two Strategies for Worst-Case Optimal Joins

The paper begins with the triangle query as a motivating example, illustrating two distinct strategies for achieving worst-case optimality. These strategies are then generalized to more complex join queries, showing how the theoretical bounds can be matched in practice by carefully designed algorithms.

This paper is start from the question of triangle query, which is defined as:

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

the simplest cyclic join, yet rich enough to illustrate most of the principles behind modern worst-case optimal join algorithms. This query can also be interpreted as enumerating all triangles in a graph $G = (V,E)$ by setting R, S, and T to contain the directed edge pairs

The traditional evaluation of $Q_\triangle$ using pair-wise join plans and show how skew with disproportionately high degree can cause large intermediate results and quadratic runtime. They then introduce two closely related algorithmic ideas that mitigate the impact of skew in this setting: the power of two choices and delaying the computation. These strategies form the conceptual core of recent join processing algorithms, enabling them to achieve optimal $O(N^{1.5})$ performance on skewed instances where classical plans require $\Omega(N^2)$ time.

2.1.1 Algorithm1: The Power of Two Choices

Let $Q_\triangle^{a_i} = \pi_{B,C}(\sigma_{A=a_i}(Q_\triangle))$ denote the projection of the output restricted to $A = a_i$. A value a_i is deemed heavy if

$$|\sigma_{A=a_i}(R \bowtie T)| \geq |Q_\triangle^{a_i}|,$$

meaning its contribution to the intermediate join $R \bowtie T$ exceeds its contribution to the final output. Since

$$|\sigma_{A=a_i}(R \bowtie T)| = |\sigma_{A=a_i} R| \cdot |\sigma_{A=a_i} T|,$$

The left-hand side can be computed from indexes on the inputs.

For the right hand side, the exact $|Q_\triangle^{a_i}|$ is unknown a priori, but because $Q_\triangle^{a_i} \subseteq S$, $|S|$ serves as a safe proxy.

The two choices refer to two alternative ways of computing $Q_\triangle^{a_i}$:

1. Light case: Compute $\sigma_{A=a_i} R \bowtie \sigma_{A=a_i} T$ and filter results by probing S.

2. Heavy case: For each $(b, c) \in S$, check whether $(a_i, b) \in R$ and $(a_i, c) \in T$.

Light keys use option(1), heavy keys use option(2). This selective branching ensures that for each $a \in L = \pi_A(R) \cap \pi_A(T)$, the work is

$$\min\{\, |\sigma_{A=a} R| \cdot |\sigma_{A=a} T|,\ |S| \,\},$$

and the total cost is

$$\sum_{a \in L} \min\{\, |\sigma_{A=a} R| \cdot |\sigma_{A=a} T|,\ |S| \,\}.$$

Bounding $\min(x, y) \leq \sqrt{xy}$ and applying Cauchy–Schwarz yields the AGM-bound runtime $O(N^{3/2})$ when $|R| = |S| = |T| = N$.

2.1.2 Algorithm2: Delaying the Computation

The second strategy also distinguishes heavy and light keys but does so *implicitly* by progressively narrowing candidate sets, rather than classifying heaviness up front. For each $a \in L_A = \pi_A(R) \cap \pi_A(T)$, it first computes

$$L_B^a = \pi_B(\sigma_{A=a} R) \cap \pi_B(S),$$

restricting B values to those that can join with a in both R and S. For each $b \in L_B^a$, it then computes

$$L_C^{a,b} = \pi_C(\sigma_{B=b} S) \cap \pi_C(\sigma_{A=a} T),$$

further restricting C values to those that can join with both (b) in S and (a) in T. This "look deeper" approach naturally reduces skew: heavy a values are filtered down by b candidates, and heavy (a, b) pairs are filtered down by c candidates, without explicit heaviness thresholds.

This delayed computation also runs in $O(N)$ time, matching the output size, and in the general case achieves the same $O(N^{3/2})$ worst-case bound as Algorithm1.

2.2 Introduction of AGM Bounds

A second key idea underpinning worst-case optimal join algorithms is the AGM bound, this part elaborate what it is and how to calcualate it in generally.

A fractional edge cover of H is a vector $x = (x_F)_{F \in E}$ satisfying:

$$\{x \mid \sum_{F:v \in F} x_F \ \geq \ 1 \quad \forall v \in V, \quad x \geq 0\}$$

The AGM inequality (Atserias–Grohe–Marx) states that for any fractional edge cover x:

$$\mid Q \mid = \mid \bowtie_{F \in E} R_F \mid \leq \prod_{F \in E} |R_F|^{x_F}.$$

This bound is tight for many queries and instances, and the choice of x can adapt the bound to the data distribution.

Example 1: Triangle Query $Q_\triangle$

For $Q_\triangle = R(A,B) \bowtie S(B,C) \bowtie T(A,C)$, the hypergraph has $V = \{A,B,C\}$ and $E = \{\{A,B\},\{B,C\},\{A,C\}\}$.

Two valid covers:

1. Symmetric cover: $x_R = x_S = x_T = \frac{1}{2}$ Bound:

$$|Q_\triangle| \ \leq \ (|R|\,|S|\,|T|)^{1/2}.$$

If $|R| = |S| = |T| = N$, this yields $N^{3/2}$, which is tight.

2. Asymmetric cover: $x_R = 1,\ x_T = 1,\ x_S = 0$ Bound:

$$|Q_\triangle| \ \leq \ |R|\,|T|.$$

This can be tighter when S is large but R and T are small.

Example 2: Clique Query K_n

Attributes: $V = \{1,\ldots,n\}$, relations $R_{i,j}$ for each $1 \leq i < j \leq n$.

Cover: $x_{i,j} = \frac{1}{n-1}$ for all $i < j$.

Bound:

$$|K_n| \ \leq \ \prod_{i<j} |R_{i,j}|^{1/(n-1)}.$$

There are $m = \binom{n}{2}$ relationships.If all $|R_{i,j}| = N$, this simplifies to $N^{n/2}$.

Example 3: Loomis–Whitney Query LW_n

Attributes: $V = \{1,\ldots,n\}$, relations R_{-i} on $V \setminus \{i\}$ for each i.

Cover: $x_{-i} = \frac{1}{n-1}$ for all i.

Bound:

$$|LW_n| \ \leq \ \prod_{i=1}^{n} |R_{-i}|^{1/(n-1)}.$$

There are $m = n$ relationships. If all $|R_{-i}| = N$, this is $N^{1+\frac{1}{n-1}}$.

2.3 A proof of the AGM bound

A third key idea in the theory of worst-case optimal joins is a succinct inductive proof of the AGM inequality, built on a query decomposition lemma. This proof not only establishes the bound

$$|\Join_{F\in E} R_F| \ \le\ \prod_{F\in E} |R_F|^{x_F}$$

for any fractional edge cover x, but also reveals a recursive structure that directly inspires the Generic-Join algorithm.

2.3.1 Lemma4.1 (Query Decomposition)

Let $Q = \Join_{F\in E} R_F$ be a natural join query represented by a hypergraph $H = (V, E)$, and let x be any fractional edge cover of H. Partition the attributes as $V = I \cup J$ with $1 \le |I| < |V|$, and define

$$L = \Join_{F\in E_I} \pi_I(R_F),$$

Then the lemma states:

$$\sum_{t_I\in L} \prod_{F\in E_J} \left| R_F \Join t_I \right|^{x_F} \ \le\ \prod_{F\in E} |R_F|^{x_F}.$$

This inequality decomposes the global join bound into a bound on the sum over partial tuples t_I of the bounds for the residual joins on J.

2.3.2 Inductive Proof of the AGM Inequality

Base case ($|V| = 1$): Here each R_F is unary. For any fractional edge cover x,

$$|\Join_{F\in E} R_F| \ \le\ \min_{F\in E} |R_F| \ \le\ \left(\min_{F\in E} |R_F|\right)^{\sum_{F\in E} x_F} \ =\ \prod_{F\in E} \left(\min_{F'\in E} |R_{F'}|\right)^{x_F} \ \le\ \prod_{F\in E} |R_F|^{x_F}.$$

Inductive step ($|V| = n \ge 2$): Let $V = I \cup J$ be any nontrivial partition, and define $L = \Join_{F\in E_I} \pi_I\big(R_F\big)$, as in Lemma4.1. For each $t_I \in L$, define the residual join query

$$Q[t_I] := \Join_{F\in E_J} \pi_J\big(R_F \Join t_I\big).$$

The original join can be written as the disjoint union

$$Q = \bigcup_{t_I\in L} \{t_I\} \times Q[t_I].$$

By the inductive hypothesis, for each t_I,

$$|Q[t_I]| \ \le\ \prod_{F\in E_J} \left| \pi_J(R_F \Join t_I) \right|^{x_F} = \prod_{F\in E_J} |R_F \Join t_I|^{x_F}$$

Summing over all $t_I \in L$ and applying Lemma4.1 yields:

$$|Q| \ =\ \sum_{t_I\in L} |Q[t_I]| \ \le\ \sum_{t_I\in L} \prod_{F\in E_J} |R_F \Join t_I|^{x_F} \ \le\ \prod_{F\in E} |R_F|^{x_F}.$$

Q.E.D

2.4 Generic-Join and Its Instantiations

2.4.1 Generic-Join: A Recursive Worst-Case Optimal Join Framework

The Generic-Join algorithm is a direct procedural embodiment of the inductive proof of the AGM bound. It evaluates a natural join query by recursively partitioning the attribute set, computing partial joins on one part, and extending them with joins on the remaining attributes. This structure ensures that the runtime matches the AGM bound in the worst case.

```
Algorithm3: Generic-Join($_{FE}R_F$)
$Input: QueryQ, hypergraphH = (V, E)$
Q = $\varnothing$
If $|V| = 1$ then
    return $\bigcap_{F \in E} R_F$
Pick $I$ arbitrarily such that $1 \leq |I| < |V|$
$L =$ Generic-Join $(\bowtie_{F \in E_I} \pi_I(R_F))$
For each $t_I \in L$ do
    $Q[t_I] =$ Generic-Join $(\bowtie_{F \in E_J} \pi_J(R_F \bowtie t_I))$
    $Q = Q \cup (\{t_I\} \times Q[t_I])$
Return $Q$
```

Here: - E_I are the edges with attributes in I, - E_J are the edges with attributes in $J = V \setminus I$, - π_I and π_J are projections onto I and J respectively.

Runtime guarantee: With pre-indexed inputs, the base case runs in $\tilde{O}(m \cdot \min_F |R_F|)$, and by induction the total runtime is

$$\tilde{O}\left(mn \cdot \prod_{F \in E} |R_F|^{x_F}\right),$$

matching the AGM bound up to polylogarithmic factors.

2.4.2 NPRR Algorithm

The NPRR algorithm is a specific instantiation of the Generic-Join framework:

- Chooses J to be a single hyperedge from E, with $I = V \setminus J$.
- Solves subqueries $Q[t_I]$ using the *power of two choices* strategy:
    1. For each partial tuple t_I, either:
        (a) Iterate over tuples in R_J and check membership in the other projected joins, or
        (b) Compute the join of the other projections first and then check membership in R_J.
    2. The choice is made to minimize work, based on comparing $|R_J|$ and the product of projected sizes.
- This dynamic choice ensures the per-t_I work is bounded by the product term in the AGM bound, yielding worst-case optimality.

2.4.3 Leapfrog Triejoin

The Leapfrog Triejoin algorithm is another instantiation of Generic-Join:

- Fixes a global attribute order and sets I to be all but the last attribute (or equivalently, recurses one attribute at a time).
- Uses trie-like iterators over sorted indexes to perform incremental intersection of candidate values for each attribute in sequence.
- At each level, restricts the domain of the next attribute by intersecting projections from all relevant relations, thereby implicitly handling skew without explicit heavy/light classification.
- This attribute-at-a-time traversal matches the recursive structure of Generic-Join and achieves the AGM-bound runtime.

3 Comparison with Prior Work

When this survey was published, the problem of evaluating relational joins had already been explored for decades. Classic algorithms and commercial database engines had refined them with sophisticated cost-based optimizers. Despite this long history, the textbook approach to join processing was still fundamentally suboptimal in the worst case.

Earlier work on acyclic queries and bounded treewidth/graphwidth focused on structural properties or just the relation cardinalities. This paper indicated that AGM's fractional edge cover number incorporates both structure and sizes, yielding tighter bounds.

Traditional optimizers rely on binary joins and heuristics for skew, without worst-case guarantees. The reviewed algorithms break from this paradigm, evaluating joins in a multiway, attribute-at-a-time fashion.

AGM provided the bound and join-project plans; this paper offers a more accessible proof, algorithmic interpretations, and a unifying framework for NPRR and Leapfrog Triejoin.

4 Evaluation Approach

No empirical system-level benchmarks are provided; the focus is on formal guarantees. The paper's evaluation is almost entirely theoretical, centering on formal proofs and complexity analysis rather than empirical benchmarks. It establishes the AGM bound through an inductive argument built on the query decomposition lemma and Hölder's inequality.

5 Key Takeaways

5.1 Strengths

1. Presents an unified framework (Generic-Join) encompassing multiple worst-case optimal algorithms. This unified view aids teaching and dissemination of worst-case optimal

join concepts.
2. Simplifies AGM's original entropy-based proof into an inductive argument accessible to a broader audience.
3. Clearly delineates the limitations of traditional join plans, motivating the adoption of new algorithms.
4. Pointing out the shortcomings of the original method is due to skew. It combines the method both in structure and cardinality way.

5.2 Weaknesses

1. Theoretical optimality does not guarantee engineering optimality; But there is no experimental results to assess practical performance in real systems.
2. The transition between specific algorithm examples (Section 2) and the general AGM bound (Section 3) is abrupt; the relationship could be made more explicit.
3. Some formulas have problems.

6 Detailed Comments

6.1 Further Discussion on Strengths and Weaknesses

One of the most valuable contributions of the paper is its ability to unify multiple worst-case optimal join algorithms under the Generic-Join framework. This is not just a theoretical convenience — it creates a common language for researchers and practitioners, making it easier to compare, teach, and extend these methods. By reframing AGM's original entropy-based proof into a clean inductive argument, the authors lower the barrier for understanding, allowing a broader audience to grasp the deep connection between join size bounds and algorithm design. The paper also does an excellent job of exposing the inherent limitations of traditional join plans, showing that their inefficiency is not incidental but structural.

While the theoretical contributions are strong, the absence of empirical validation is a significant limitation. Worst-case optimality is a powerful guarantee, but it does not automatically translate into superior performance in real-world systems, where constant factors, cache behavior, indexing overhead, and parallel execution strategies can dominate. Without experiments, it is impossible to assess whether the proposed algorithms outperform finely tuned traditional joins on typical workloads, or whether their benefits only appear in contrived worst-case instances.

One well-known example comes from early experiments with worst-case optimal join algorithms like NPRR and Leapfrog Triejoin in distributed systems such as Spark or parallel relational engines. On paper, these algorithms beat traditional pairwise joins for cyclic queries (e.g., triangle enumeration) because their runtime matches the AGM bound. But in practice, researchers found that on moderate-sized, real-world datasets, a well-tuned hash join pipeline could outperform them. Not because the theory was wrong, but because:

1. Indexing overhead: Worst-case optimal joins often require building multi-attribute indexes or trie structures before execution. On small or skew-light datasets, this preprocessing cost can dominate.

2. Data characteristics: On many real graphs, the degree distribution is far from the worst-case skew that these algorithms are designed to handle. In such cases, the "extra"

machinery for skew-resilience doesn't pay off.

The paper's structure also leaves room for improvement. The jump from concrete algorithmic examples in Section2 to the abstract AGM bound in Section3 feels abrupt, with little narrative to guide the reader through how the earlier examples motivate the general theory.

Finally, the presentation of some formulas is problematic. In Section2.2, the paper presents the formula

$$|\sigma_{A=a_i}(R \bowtie S)| = |\sigma_{A=a_i}R| \cdot |\sigma_{A=a_i}S|.$$

However, this may be a typo. Since relation S does not contain attribute A, the correct expression should be

$$|\sigma_{A=a_i}(R \bowtie T)| = |\sigma_{A=a_i}R| \cdot |\sigma_{A=a_i}T|.$$

Although this is a minor slip, it can cause confusion for readers, especially those trying to follow the derivation step-by-step. Clarifying this point would improve the paper's precision and readability.

6.2 Relation to My Research

The AGM bound and the Generic-Join framework offer principled ways to reason about multi-relation joins, which are a core operation in large-scale retrieval pipelines when combining heterogeneous metadata, annotations, and content features. The skew-handling strategies described here could inform how such a system efficiently fuses high-dimensional embeddings with structured relational data, ensuring that retrieval remains efficient even in worst-case or highly imbalanced scenarios.

6.3 Future Directions

1. Develop algorithms that adapt to the actual input distribution while retaining worst-case guarantees.
2. Combine worst-case optimal joins with vectorized execution, column stores, and parallel/distributed architectures.
3. Incorporate complex functional dependencies and other integrity constraints into the bound and algorithms.

7 Reference

Ngo, H. Q., Ré, C., Rudra, A. (2014). Skew strikes back: new developments in the theory of join algorithms. Acm Sigmod Record, 42(4), 5-16.