Instance-Optimal Acyclic Joins: From Theory to Systems

Paraschos Koutris
University of Wisconsin–Madison
USA
paris@cs.wisc.edu

Stijn Vansummeren
UHasselt, Data Science Institute
Belgium
stijn.vansummeren@uhasselt.be

Qichen Wang
Nanyang Technological University
Singapore
qichen.wang@ntu.edu.sg

Yisu Remy Wang
University of California, Los Angeles
USA
remywang@cs.ucla.edu

Xiangyao Yu
University of Wisconsin–Madison
USA
yxy@cs.wisc.edu

ABSTRACT

This tutorial revisits Yannakakis algorithm (YA) as a central example of how database theory can guide practical query processing. Yannakakis algorithm gives an instance-optimal guarantee for evaluating acyclic joins, and its core ideas, including join trees, semijoin reduction, and information passing, have influenced decades of work in query evaluation. Recent studies have renewed interest in the Yannakakis algorithm by demonstrating that its structure-aware principles can be applied to modern database systems, yielding practical methods for robust SQL analytics. This tutorial introduces the foundations of the Yannakakis algorithm and acyclic query processing, surveys recent advances in theory and systems, and discusses how these ideas extend to query optimization and general queries beyond the multi-way joins. The goal is to give the audience both a clear conceptual understanding of Yannakakis algorithm and a broad view of its growing role in modern data management.

PVLDB Reference Format:
Paraschos Koutris, Stijn Vansummeren, Qichen Wang, Yisu Remy Wang, and Xiangyao Yu. Instance-Optimal Acyclic Joins: From Theory to Systems. PVLDB, 19(12): 4884 - 4887, 2026.
doi:10.14778/3827998.3828144

1 INTRODUCTION

In 1981, an optimal algorithm for computing acyclic joins was concurrently discovered by Bernstein et al. [4, 5] and Yannakakis [23], and has since become known as Yannakakis algorithm. The algorithm is conceptually simple, applicable to a wide class of practical queries, and has the strong guarantee of being *instance optimal*, which means it has the best possible asymptotic complexity for *any* input instance. As a result, it has shaped how we understand acyclicity, full reduction, and structure-aware query evaluation. However, despite its clean theory and strong guarantees, Yannakakis' algorithm has rarely been implemented in mainstream database systems, presumably due to the rather large overhead incurred by naive, straightforward implementation methods. For instance, recent experiments by Gottlob et al. [11] demonstrate that while the Yannakakis algorithm improves tail-end performance for long-running queries, it increases run time by an average of 2.4× compared to standard binary joins.

The crucial idea underlying Yannakakis' algorithm is to use semijoins to remove so-called *dangling tuples*: input tuples that do not contribute to any output join tuple. The algorithm works in three passes: two semijoin passes to remove dangling tuples, followed by final join pass on the reduced input to compute the actual join result. The semijoin passes in particular are required to achieve the instance-optimal asymptotic complexity. Standard binary joins, by contrast, do not remove dangling tuples and require only a single pass (the actual join), and do therefore not achieve instance-optimal aysmptotic complexity. However, if there were no or few dangling tuples to begin with, then the overhead of performing the semijoin passes may not outweigh the performance gain in the actual join pass, leading to the above-cited slowdown. Furthermore, the algorithm's three-pass design complicates its integration into existing query optimizers.

In response to these observations, recent work has started to close the gap between theory and practice in several complementary ways. The first line of work improves the efficiency of semijoin reduction itself, mainly through bitvector-based filtering techniques [9, 15, 22, 24, 25]. In particular, Predicate Transfer [22] and Robust Predicate Transfer [25], propose to use Bloom-filter-based semijoins instead of traditional hash-based semijoins to make dangling tuple pruning much cheaper while still retaining the core benefit of reducing useless tuples early. The second line of work focuses on query rewriting: since Yannakakis' algorithm can be expressed using standard relational operators, one can build an external structure-aware planner that rewrites the algorithm into ordinary SQL or a DAG-style relational plan, making deployment possible on top of existing systems [11, 18]. The third line of work [3, 6, 12] goes even further by designing Yannakakis-style execution methods that aim to achieve instance-optimality without regression relative to standard binary join processing, by merging semijoin reduction directly into join execution rather than treating it as a separate preprocessing stage. Together, these three directions show that Yannakakis' algorithm is no longer only a classical theoretical result, but a growing foundation for practical, structure-guided query processing that is ready for wide-spread systems adoption.

This tutorial aims to survey these exciting recent developments, starting with the theoretical background of Yannakis' algorithm,

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828144

and also discussing possible extensions. The tutorial is organized into four parts. We begin with the basic concepts behind Yannakakis' algorithm, including acyclicity, join trees, semijoin reduction, information passing, and full reducers, so that the later material is self-contained. We then survey the three main recent directions outlined above: more efficient filtering, rewrite-based implementations, and zero-overhead Yannakakis-style execution. After that, we discuss broader Yannakakis-style algorithms and extensions, including support for aggregation, top-k, and comparisons.

The tutorial is intended for a broad database audience, including researchers, graduate students, and practitioners working on query processing, query optimization, and the interface between database theory and systems. We assume only a standard background in relational databases and join processing. The opening part is designed to make the tutorial accessible to attendees from both systems and theory backgrounds, while the latter parts aim to give a clear picture of why Yannakakis' algorithm has become newly relevant for modern data management.

1.1 Related Tutorials

Recent tutorials have covered related topics, but with a different focus from ours. The SIGMOD 2020 tutorial "Worst Optimal Join Algorithms Meet Top-k" studied the connection between optimal join processing and top-k or ranked enumeration. The ICDE 2022 tutorial "Toward Responsive DBMS: Optimal Join Algorithms, Enumeration, Factorization, Ranking, and Dynamic Programming" extended the SIGMOD 2020 tutorial, covering responsive query answering, enumeration, factorization, ranking, and dynamic programming. In contrast, our tutorial is centered on Yannakakis' algorithm itself. We focus on why the Yannakakis algorithm is a fundamental result for acyclic joins and the methods used by recent work to prove its viability for practical integration into concrete systems such as DuckDB [18, 25], Umbra [6] and Apache Datafusion [3]. In particular, Yannakakis algorithm offers the much stronger *instance optimal* guarantee on acyclic queries as compared to the worst-case guarantee provided by worst-case optimal joins (if executed without the guidance of a tree decomposition). At the same time, our tutorial will also cover part of the above topics, but from a different angle: we present them as general techniques built on the Yannakakis algorithm, showing how it gives rise to a new paradigm for query processing based on query structure. Thus, while earlier tutorials focus broadly on top-k queries, our tutorial highlights structure-guided processing through the lens of Yannakakis' algorithm and its extensions.

2 TUTORIAL CONTENTS

2.1 Background

This part of the tutorial provides an introduction to the basic concepts of the Yannakakis algorithm, including query acyclicity, join trees, full reducers, and the classical three-phase evaluation framework. We will explain how the acyclic structure of a join query can be captured by a join tree, and how this structure enables efficient information passing across relations via semijoin reductions before the final join. This naturally leads to the notion of a full reducer, where all remaining tuples are guaranteed to contribute to some output. The goal of this part is to give the audience an intuitive yet technically clear understanding of why Yannakakis algorithm is instance-optimal and how it relates to query structure, and to prepare them for the later parts of the tutorial on recent improvements and generalizations.

2.2 Improvements of Yannakakis Algorithm

This part of the tutorial covers recent progress on making Yannakakis' algorithm practical. It is organized around three themes: improving semijoin efficiency with bitvector filters, unlocking the Yannakakis algorithm through query rewriting, and achieving optimality without regression relative to standard binary joins. These three directions reflect different ways of reducing the overhead that has historically prevented the Yannakakis algorithm from being adopted in mainstream systems.

Improving semijoin efficiency with Bitvector filters. First, we will discuss work that improves the efficiency of semijoin reduction by using bitvector-based filters such as Bloom filters. The key idea is to use lightweight pre-filtering to remove tuples that cannot contribute to the final join result, while keeping memory usage low and improving cache behavior. In this part, we will cover Predicate Transfer [22] and its extension Robust Predicate Transfer [25], which demonstrate how Bloom-filter-style information passing can improve robustness and, in the case of RPT, recover the theoretical guarantees of Yannakakis algorithm for acyclic queries while better aligning with a modern engine.

Unlocking theoretical efficiency with query rewrite. Second, we will cover query rewriting as a portable way to realize Yannakakis-style evaluation on top of existing database systems. Since Yannakakis algorithm can be expressed using standard relational operators, an external planner can generate a structure-aware plan and translate it into SQL or a DAG-style relational plan for execution by an unmodified DBMS [11]. We will use this part to explain why rewriting is attractive for deployment, how recent work, such as Yannakakis+ [18], reduces the number of semijoin passes and leverages join-tree-aware optimization, and how the same framework can be extended to support aggregation [16, 18].

Achieving Optimality without Regression. We will introduce a newer line of work that aims to obtain the theoretical benefits of Yannakakis algorithm with zero overhead relative to standard binary hash joins. The main idea is to piggyback semijoin reduction directly into join execution, either during hash-table construction in bottom-up methods [3, 6] or during probing in top-down methods [12]. We will present this line as an important step beyond merely reducing Yannakakis algorithm's overhead: it suggests that Yannakakis-style processing can serve as the basis for a new practical paradigm of query execution, where structure-aware pruning is built into the join operators themselves rather than added as a separate preprocessing stage.

2.3 Extensions for General Query Evaluation

This part of the tutorial discusses how the high-level idea behind Yannakakis algorithm can be extended beyond plain acyclic join evaluation into a more general *reduction-enumeration* framework for query answering. The main idea is that the information-passing

view of the Yannakakis algorithm is not limited to classical semi-join reduction: one can first compute a partial reducer and reduce the query with one less relation, guided by the join-tree structure that preserves all information needed for the final answers, and then run an enumeration phase that reconstructs the output efficiently. This viewpoint goes back to work on constant-delay enumeration for acyclic conjunctive queries [1], and has since been developed in several directions, including comparisons [14, 21], rank enumeration/top-k [7, 8, 17, 20], dynamic evaluation [13, 19], and sampling [2, 10, 26]. In many of these settings, the reduction phase can still be carried out in linear or near-linear time, while the enumeration phase provides output-sensitive access to the query answers, making the overall approach instance-optimal or near-instance-optimal for the corresponding task.

In particular, we will focus on the general reduction-enumeration idea and use comparisons as the main example of how Yannakakis-style information passing can be extended beyond pure equi-joins. This includes recent work showing that suitable acyclic conditions still permit linear-time evaluation for conjunctive queries with comparisons.

2.4 Open Challenges and Future Work

Finally, we close the tutorial with an overview of the challenges that remain in bridging theory and practice for structure-guided query evaluation in general, and the corresponding opportunities for future work.

3 PRESENTERS

Paraschos Koutris is an associate professor in Computer Sciences at the University of Wisconsin-Madison. His research lies in the intersection of data management theory and practice, focusing on data processing for massively parallel systems, uncertain data, and join processing. He has won the SIGMOD Jim Gray Dissertation Award for his work on the foundations of parallel data processing and received the PODS 2023 Test-of-Time award, as well as multiple Distinguished Paper awards from SIGMOD and PODS.

Stijn Vansummeren is Research Professor at the Data Science Institute of Hasselt University, Belgium where he is embedded in the Data-Intensive Computing (DINCOMP) research group. He has previously given tutorials at DEBS 2017 and CIKM 2018. His work on Dynamic Yannakakis received the ACM SIGMOD 2018 Research Highlights Award.

Qichen Wang is an Assistant Professor at the College of Computing and Data Science, Nanyang Technological University, Singapore. He is working on query optimization, dynamic query evaluation, and join algorithms. His work on comparison queries received the ACM SIGMOD 2022 Best Paper Honorable Mention award.

Yisu Remy Wang is an Assistant Professor at the Computer Science Department at UCLA. His work on join algorithms and query optimization has received numerous awards from SIGMOD, PODS, OOPSLA, and POPL.

Xiangyao Yu is an Assistant Professor in Computer Sciences at the University of Wisconsin-Madison, working on database systems. He has received the NSF CAREER Award, the Sloan Research Fellowship, and the VLDB Early Career Research Contribution Award.

ACKNOWLEDGMENTS

Qichen Wang was supported by the Singapore Ministry of Education under Grant No. RS32/25 and Nanyang Technological University Global Research Excellence Award for Travel (GREAT). Stijn Vansummeren was supported by the Bijzonder Onderzoeksfonds (BOF) of Hasselt University (Belgium) under Grant No. BOF20ZAP02 and by the Research Foundation Flanders (FWO) under Grant No. G0B9623N.

REFERENCES

[1] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. 2007. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings (Lecture Notes in Computer Science)*, Jacques Duparc and Thomas A. Henzinger (Eds.), Vol. 4646. Springer, 208–222. https://doi.org/10.1007/978-3-540-74915-8_18

[2] Liese Bekker, Lorrens Pantelis, Frank Neven, and Stijn Vansummeren. 2026. Poisson Sampling over Acyclic Joins. *Proc. ACM Manag. Data* 4, 2 (SIGMOD), Article 224 (2026). https://doi.org/10.1145/3802101

[3] Liese Bekkers, Frank Neven, Stijn Vansummeren, and Yisu Remy Wang. 2025. Instance-Optimal Acyclic Join Processing Without Regret: Engineering the Yannakakis Algorithm in Column Stores. *Proc. VLDB Endow.* 18, 8 (2025), 2413–2426. https://www.vldb.org/pvldb/vol18/p2413-vansummeren.pdf

[4] Philip A. Bernstein and Dah-Ming W. Chiu. 1981. Using Semi-Joins to Solve Relational Queries. *J. ACM* 28, 1 (1981), 25–40. https://doi.org/10.1145/322234.322238

[5] Philip A. Bernstein, Nathan Goodman, Eugene Wong, Christopher L. Reeve, and James B. Rothnie Jr. 1981. Query Processing in a System for Distributed Databases (SDD-1). *ACM Trans. Database Syst.* 6, 4 (1981), 602–625. https://doi.org/10.1145/319628.319650

[6] Altan Birler, Alfons Kemper, and Thomas Neumann. 2024. Robust Join Processing with Diamond Hardened Joins. *Proc. VLDB Endow.* 17, 11 (2024), 3215–3228. https://doi.org/10.14778/3681954.3681995

[7] Shaleen Deep, Xiao Hu, and Paraschos Koutris. 2022. Ranked Enumeration of Join Queries with Projections. *Proc. VLDB Endow.* 15, 5 (jan 2022), 1024–1037. https://doi.org/10.14778/3510397.3510401

[8] Shaleen Deep and Paraschos Koutris. 2021. Ranked Enumeration of Conjunctive Query Results. In *24th International Conference on Database Theory (ICDT 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[9] Bailu Ding, Surajit Chaudhuri, and Vivek Narasayya. 2020. Bitvector-aware Query Optimization for Decision Support Queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2011–2026. https://doi.org/10.1145/3318464.3389769

[10] Aryan Esmailpour, Xiao Hu, Jinchao Huang, and Stavros Sintos. 2026. Subset Sampling over Joins. *Proc. ACM Manag. Data* 4, 2 (PODS), Article 117 (2026). https://doi.org/10.1145/3801913

[11] Georg Gottlob, Matthias Lanzinger, Davide Mario Longo, Cem Okulmus, Reinhard Pichler, and Alexander Selzer. 2023. Structure-guided query evaluation: Towards bridging the gap from theory to practice. *arXiv preprint arXiv:2303.02723* (2023).

[12] Zeyuan Hu, Yisu Remy Wang, and Daniel P Miranker. 2024. TreeTracker Join: Simple, Optimal, Fast. *arXiv preprint arXiv:2403.01631* (2024).

[13] Muhammad Idris, Martin Ugarte, and Stijn Vansummeren. 2017. The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1259–1274. https://doi.org/10.1145/3035918.3064027

[14] Muhammad Idris, Martín Ugarte, Stijn Vansummeren, Hannes Voigt, and Wolfgang Lehner. 2020. General dynamic Yannakakis: conjunctive queries with theta joins under updates. *VLDB J.* 29, 2-3 (2020), 619–653. https://doi.org/10.1007/S00778-019-00590-9

[15] Harald Lang, Thomas Neumann, Alfons Kemper, and Peter Boncz. 2019. Performance-optimal filtering: Bloom overtakes Cuckoo at high throughput. *Proc. VLDB Endow.* 12, 5 (Jan. 2019), 502–515. https://doi.org/10.14778/3303753.3303757

[16] Matthias Lanzinger, Reinhard Pichler, and Alexander Selzer. 2025. Avoiding Materialisation for Guarded Aggregate Queries. *Proc. VLDB Endow.* 18, 5 (Aug. 2025), 1398–1411. https://doi.org/10.14778/3718057.3718068

[17] Nikolaos Tziavelis, Deepak Ajwani, Wolfgang Gatterbauer, Mirek Riedewald, and Xiaofeng Yang. 2020. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. *Proc. VLDB Endow.* 13, 9 (may 2020), 1582–1597. https://doi.org/10.14778/3397230.3397250

[18] Qichen Wang, Bingnan Chen, Binyang Dai, Ke Yi, Feifei Li, and Liang Lin. 2025. Yannakakis+: Practical Acyclic Query Evaluation with Theoretical Guarantees. *Proc. ACM Manag. Data* 3, 3, Article 235 (June 2025), 28 pages. https://doi.org/10.1145/3725423

[19] Qichen Wang, Xiao Hu, Binyang Dai, and Ke Yi. 2023. Change Propagation Without Joins. *Proc. VLDB Endow.* 16, 5 (jan 2023), 1046–1058. https://doi.org/10.14778/3579075.3579080

[20] Qichen Wang, Qiyao Luo, and Yilei Wang. 2024. Relational Algorithms for Top-k Query Evaluation. *Proc. ACM Manag. Data* 2, 3, Article 168 (May 2024), 27 pages. https://doi.org/10.1145/3654971

[21] Qichen Wang and Ke Yi. 2026. Conjunctive Queries with Comparisons. *ACM Trans. Database Syst.* 51, 2, Article 7 (March 2026), 37 pages. https://doi.org/10.1145/3769424

[22] Yifei Yang, Hangdong Zhao, Xiangyao Yu, and Paraschos Koutris. 2024. Predicate Transfer: Efficient Pre-Filtering on Multi-Join Queries. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org. https://www.cidrdb.org/cidr2024/papers/p22-yang.pdf

[23] Mihalis Yannakakis. 1981. Algorithms for Acyclic Database Schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*. IEEE Computer Society, 82–94.

[24] Tim Zeyl, Qi Cheng, Reza Pournaghi, Jason Lam, Weicheng Wang, Calvin Wong, Chong Chen, and Per-Ake Larson. 2025. Including Bloom Filters in Bottom-up Optimization. In *Companion of the 2025 International Conference on Management of Data* (Berlin, Germany) *(SIGMOD/PODS '25)*. Association for Computing Machinery, New York, NY, USA, 703–715. https://doi.org/10.1145/3722212.3724440

[25] Junyi Zhao, Kai Su, Yifei Yang, Xiangyao Yu, Paraschos Koutris, and Huanchen Zhang. 2025. Debunking the Myth of Join Ordering: Toward Robust SQL Analytics. 3, 3, Article 146 (June 2025), 28 pages. https://doi.org/10.1145/3725283

[26] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random Sampling over Joins Revisited. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1525–1539. https://doi.org/10.1145/3183713.3183739