

HomeRun

Citation for published version (APA):

van Leeuwen, W., Fletcher, G., & Yakovets, N. (2024). HomeRun: A Cardinality Estimation Advisor for Graph Databases. In O. Hartig, & Z. Kaoudi (Eds.), *GRADES-NDA '24: Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)* Article 6 Association for Computing Machinery, Inc.. <https://doi.org/10.1145/3661304.3661902>

Document license:

CC BY-NC-SA

DOI:

[10.1145/3661304.3661902](https://doi.org/10.1145/3661304.3661902)

Document status and date:

Published: 09/06/2024

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.



HOMERUN: A Cardinality Estimation Advisor for Graph Databases

Wilco van Leeuwen, George Fletcher, Nikolay Yakovets

{w.j.v.leeuwen,g.h.l.fletcher,hush}@tue.nl

Eindhoven University of Technology

Eindhoven, Noord-Brabant, The Netherlands

ABSTRACT

Database systems depend on cardinality estimates for generation of optimal query execution plans. Selecting an appropriate cardinality estimation technique involves navigating trade-offs, including the accuracy of estimates, time required for estimation, and necessary statistics. These trade-offs can lead to different choices based on the dataset and query workload. Unfortunately there is limited support for advising graph database users in exploring these trade-offs and making the right choices for their scenarios. To address this critical gap, we introduce an advisor tool, HOMERUN, which analyzes the performance of various cardinality estimation techniques in given usage scenarios. We explain HOMERUN's capabilities using the industry-standard LSQB benchmark and synthetic scenarios. HOMERUN reveals how minor changes in the dataset can significantly impact the conclusions about the performance of cardinality estimation techniques.

ACM Reference Format:

Wilco van Leeuwen, George Fletcher, Nikolay Yakovets. 2024. HOMERUN: A Cardinality Estimation Advisor for Graph Databases. In *7th Joint Workshop on Graph Data Management Experiences Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA '24)*, June 14, 2024, Santiago, AA, Chile. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3661304.3661902>

1 INTRODUCTION

Accurate and efficient cardinality estimation (CE) is crucial for effective query planning of declarative queries. Graph databases poses serious challenges on cardinality estimation due to their flexibility and missing schema information. Numerous CE techniques have been developed, each with its own advantages and disadvantages. This work focuses on understanding the appropriate circumstances for using different CE techniques.

Generally, a CE technique should provide accurate estimates while minimizing resource usage. We categorize resource usage into two types: 1) resource usage at *estimation* time and 2) resource usage at *preparation* time. Estimation time refers to the online phase when a (sub)query expression is presented to a dataset, and a cardinality estimate is required. Preparation time is the offline phase when a new or updated dataset is being prepared for use. This preparation may involve obtaining a sample, computing statistics, or preparing a summary of the dataset.

The effectiveness of CE techniques is highly dependent on the usage scenario, i.e. the dataset and query workload. For example,

with simple datasets and straightforward query workloads, simple CE techniques that deliver very fast (though possibly less accurate) estimates can outperform more sophisticated techniques. However, for larger datasets or more complex queries, the more advanced CE techniques may lead to significantly better overall end-to-end (E2E) performance. This is because the negative impact of choosing a poor execution plan can outweigh the extra time spent on CE during query planning. Here, E2E performance refers to resource usage from the moment the input query is received until the final result is returned to the user, encompassing both the planning and execution stages.

However, estimating the performance of CE techniques for a specific scenario is often not straightforward. In fact (and as we will show in Section 5), even minor changes in the dataset can lead to entirely different outcomes. A technique may excel in one scenario, but minor alterations in the dataset could lead to poor performance (and vice versa for different techniques).

This variability makes it challenging to provide definitive guidelines on which CE technique to use in any given scenario. To address this, we have developed an advisor tool, HOMERUN, that analyzes a specific scenario and offers performance insights and recommendations. Section 3 introduces all the elements of this advisor tool. Section 4 illustrates the application of our advisor tool on the LSQB scenario demonstrating its utility in practice. Lastly, Section 5 shows how minor changes in the dataset can significantly impact the performance of CE techniques.

2 RELATED WORK

Cardinality estimation has become a fundamental part of Database Management Systems (DBMSs) since Selinger et al. [16] described their approach to query optimization in System R. Since then, numerous cardinality estimation techniques have been developed, including Histograms [15], Sampling [10, 12, 18], Summarization [17], Synopsis-based methods [2, 3, 14], and ML-based approaches [5, 19, 20].

Early comparison studies used *TPC-H* as the dataset and query workload. However, the data generators of *TPC-H* use uniform distributions and lack complicated dependencies between values in different attributes to simplify data generation and scalability. As a result, cardinality estimation techniques that rely on these simplifying assumptions may display overly optimistic estimation accuracy. Leis et al. [9, 11] identified this issue and proposed using a real instance of the *IMDB* database as the dataset, which features complex correlations and non-uniform distributions. They hand-crafted a set of queries reflecting potential queries by movie enthusiasts. Their benchmark, named the *Join Order Benchmark* (JOB), includes many complex predicates, such as the *SQL LIKE* operator on strings and disjunctions. *JOB-light* [6] was later introduced to focus only



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

GRADES-NDA '24, June 14, 2024, Santiago, AA, Chile

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0653-0/24/06

<https://doi.org/10.1145/3661304.3661902>

on predicates involving numerical and categorical attributes, allowing fair comparisons of techniques that cannot estimate complex predicates. Later, the small join graph of JOB inspired Han et al. [4] to introduce a new benchmark based on real data and a handpicked query workload, named *STATS-CEB*. Their dataset is an anonymized dump of user-contributed content from the Stats Stack Exchange network.

LDBC SNB offers a scalable synthetic data generator that produces networks with a predefined schema, featuring distributions and correlations akin to those in real social networks [1]. It encompasses an Interactive and a Business Intelligence workload. The Interactive workload includes user-centric transactional-like interactive queries, while the Business Intelligence workload is designed for analytic queries addressing business-critical questions. The *LSQB* benchmark [13] utilizes the *LDBC SNB* data generator but focuses on a query workload comprising labelled subgraph matching queries.

The work described in this paper stands apart from previous efforts by introducing a tool to analyze the impact of various cardinality estimation (CE) techniques in a user-specified scenario, involving both a dataset and a query workload. It seeks to determine the most suitable CE technique for a specific, user-defined scenario, rather than for a general scenario set by a benchmark. This tool is demonstrated using the *LSQB* benchmark scenario, along with several synthetic scenarios specifically crafted to emphasize how the performance of CE techniques depends on the exact scenario.

3 ADVISOR TOOL

The primary objective of *HOMERUN* is to facilitate the comparison of various CE techniques across different scenarios. Our advisor tool requires two main inputs: (1) *dataset and query set*: the tool takes a specific dataset and a set of queries to be executed on this dataset. This input allows the tool to evaluate the performance of CE techniques in the context of real-world data and query patterns; and, (2) *cardinality estimation techniques for comparison*: the tool also requires the selection of CE techniques to be compared. For this purpose, we use a recently proposed CE framework [7] with various configurations. These configurations consist of a combination of Partial Estimation Techniques (PETs) and Extend Partial Estimation Set Techniques (EPESTs). For Combination Techniques, we always use the approach based on Conditional Independence assumptions.

3.1 Visualizing QEP execution time and planning time

A query can have multiple Query Execution Plans (QEPs), each with its unique characteristics in terms of resource usage when evaluated on a dataset. The primary resource of interest here is the execution time. Each cardinality estimation (CE) technique will select a QEP for each query. For each QEP chosen by a CE technique, two key metrics will be visualized: *execution time* (*execTime*) and *planning time* (*planningTime*). The sum of these two metrics will represent the end-to-end (E2E) running time of the query.

An illustrative example can be seen in Figure 1a. In this figure, 12 different CE techniques were evaluated (including C2, PC2, ..., CS). Eight of these techniques selected plan *p14*, which has an *execTime*

of approximately 2 milliseconds, while the others chose a more expensive plan *p5*, with an *execTime* of 29 milliseconds. The yellow bar in the figure represents the *execTime* for each QEP. The bars on top of each yellow bar indicate the *planningTime* for the CE technique that selected that particular QEP.

This visualization approach provides a clear and concise way to compare the performance of different CE techniques, focusing on both the execution efficiency of the chosen QEP and the efficiency of the planning process itself.

3.2 Visualizing the entire plan space

If the number of Query Execution Plans (QEPs) is manageable and each can be executed within a reasonable timeframe, the visualization method can be expanded to display the execution time of every QEP. In this extended visualization, a small plot will be included to represent each QEP's execution time with a dot. The QEPs that are particularly important or interesting – such as the fastest, the slowest, and those selected by various cardinality estimation (CE) techniques – will be highlighted in black on this plot. The remaining QEPs, which are less critical for the immediate analysis, will be marked in gray. This method of categorization helps in quickly identifying the most relevant QEPs for detailed analysis while still providing a comprehensive overview of all available QEPs. A yellow bar without bars on top represent QEPs that have not been chosen by any of the considered CE techniques.

An example of this extended visualization is shown in Figure 1b. This figure demonstrates how the visualization effectively communicates the performance of each QEP, allowing users to easily compare the execution times and discern patterns or outliers. By marking the QEPs chosen by different CE techniques in black (over gray), it becomes easy to see which techniques tend toward more efficient or less efficient plans, thus providing valuable insights for optimizing query performance.

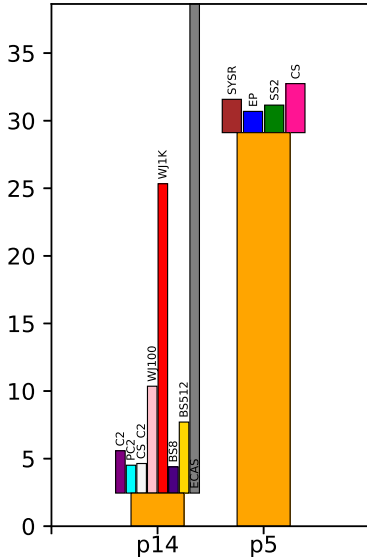
3.3 Visualizing QEPs

A QEP is visualized as a tree, where each node represents a physical operator used in the execution of the query. The details provided for each node in this tree include: (1) *type of operator*: this could be HashJoin, LabelAccess, etc.; (2) *operator parameters*: these are specific to the operator, such as *src(%8)=trg(%9)* or *(%8, 'likes')*; (3) *actual cardinality*; (4) *median execution time*; and, (5) *cardinality and cost estimates*.

All cardinality (estimates) and cost (estimates) are relative to the subquery where the current node is the root. This approach enables an understanding of how each CE technique evaluates different parts of the query, providing valuable insights into the effectiveness and rationale behind each technique's decision-making process. By comparing the estimated values against the actual cardinality and execution times, users can gauge the accuracy and efficiency of different CE techniques in real-world scenarios.

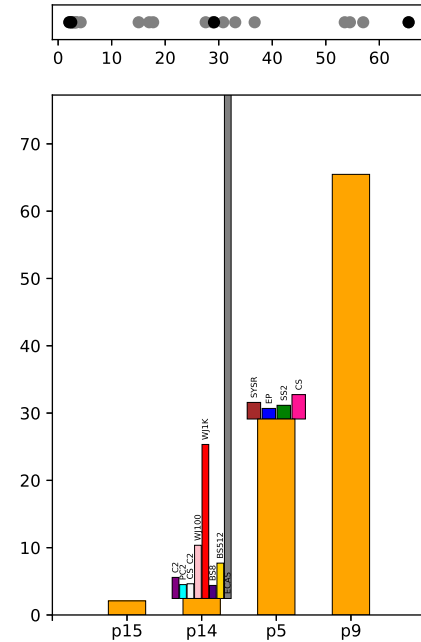
3.3.1 Note on cost estimation. The cost for different operators in the QEPs needs to be estimated such that the estimated cost positively correlates with the actual cost of executing the QEPs. We use simple formulas based on the type of operator and the input cardinalities. For access operations, the cost estimate is equal to the cardinality estimate. For indexed nested loop join (INLJ) operators,

QEP execution times and planning times (in ms)



(a) ExecTime of the QEPs that were chosen by CE techniques (yellow bars), which CE technique choose which QEP and the planningTimes (bars on top of the yellow bars)

QEP execution times for the whole plan space and planning times



(b) Extension of Figure 1a with a plot on top with the execTimes of all QEPs. The bottom plot shows the chosen QEPs, the fastest and the slowest.

Figure 1: Visualizing QEP execTimes and planningTimes

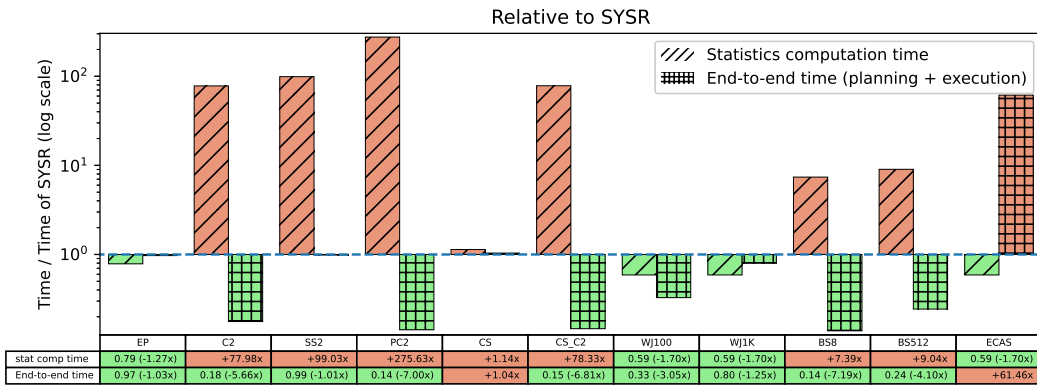


Figure 2: Relative comparison of different CE techniques (EP, C2, ...) w.r.t. a base CE techniques (SYSR) on different metrics (SCT and E2E time)

the cost estimate is the sum of the cost of its children plus the cardinality of the left child. For hash join (HJ) operators, the cost estimate is the sum of the cost of its children plus the cardinality of its children, with a preference for INLJ over HJ. All other unary operators take the cost of their child.

3.3.2 Effect of the Cost Estimation Model. To evaluate the impact of the cost estimation model, we include a cardinality estimation (CE) technique known as Exact Cardinalities for All Subexpressions

(ECAS). If ECAS fails to select the best plan, or a plan whose performance is close to the best, it suggests a flaw in the cost estimation model. Figure 1b demonstrates the effectiveness of our cost estimation model in a specific test case, as ECAS selected a plan whose performance closely mirrors that of the best plan.

Other CE techniques aim to find the same plan as ECAS. If a CE technique selects a plan that outperforms the one chosen by ECAS, it can typically be attributed to 'luck' rather than the technique's inherent performance.

3.4 Aggregating to compare different CE techniques

Showing different metrics in separate plots is an effective way to compare various cardinality estimation (CE) techniques with respect to a specific metric. However, this approach can make it challenging to compare techniques across multiple metrics simultaneously.

To address this, a comprehensive plot will be introduced that combines the values of each metric for every CE technique. This requires aggregating per-query metrics (E2E time) across all queries in the workload. The aggregated value provides a metric for the performance of each CE technique over the entire query workload, enabling a quick comparison to determine which techniques are most effective overall. In addition to these performance metrics, other important factors include the time taken to compute the required statistics for each CE technique and the storage space needed for these statistics.

Figure 2 demonstrates an effective method for comparing various cardinality estimation (CE) techniques relative to each other across different metrics.¹ The approach involves using one technique as a baseline against which all others are compared. The comparison reveals whether other techniques show improvements or regressions with respect to the baseline across different metrics. Values below 1 are colored green, indicating that they represent less time or memory usage than the baseline technique. Longer green bars (i.e., lower values) signify larger improvements.

From Figure 2, it's clear that the CE techniques EP and WJ demonstrate improvements in all metrics and, therefore, would be recommended over the baseline SYSR technique for the tested scenario. Conversely, the CS technique performs worse in all metrics. Other techniques show trade-offs, improving in one metric while regressing in another, all relative to SYSR.

The table at the bottom of the figure provides explicit values for each comparison (time/time of base, or memory/memory of base), with green and red backgrounds used to indicate improvements and regressions, respectively. For improvements, the figure displays $1/v$ in parentheses, indicating the factor by which the technique is better compared to the base CE technique, where v is the computed value.

4 USAGE OF THE ADVISOR TOOL

Figure 3 presents a series of screenshots from the front-end of our advisor tool. The tool's back-end is developed in an open-source graph database AvantGraph [8], which integrates the recently proposed framework for cardinality estimation [7]. Each cardinality estimation (CE) technique necessitates a specific configuration, comprising a set of Partial Estimation Techniques (PETs) and a set of Extend Partial Estimation Set Techniques (EPESTs). For Combination Techniques, we consistently adopt an approach based on Conditional Independence assumptions. In this document, as we are solely focusing on labelled topological patterns, each configuration will exclude the use of any EPESTs.

All experiments were conducted with the back-end running on an Ubuntu 20.04.5 LTS server, equipped with 194GB of RAM and

¹Only two metrics are displayed (E2E time and SCT) to maintain clarity and simplicity for presentation purposes

CE technique	PETs	CE technique	PETs
SYSR	labPC, sysR	CS_C2	labPC, EP, CS, CHAIN2
EP	labPC, EP	WJ100	WJ100
C2	labPC, EP, CHAIN2	WJ1K	WJ1K
SS2	labPC, EP, SSTAR2	BS8	BS8
PC2	labPC, EP, CHAIN2, SSTAR2, TSTAR2	BS512	BS512
CS	labPC, EP, CS	ECAS	ECAS

Table 1: CE techniques and PETs used in the experiments

an Intel Xeon CPU @ 3.07GHz using 24 threads. Avantgraph uses all available threads for the execution of Query Execution Plans (QEPs), although it does not yet employ multithreading for the planning stage.

4.1 CE Techniques to Compare

The **labPC** technique stores the cardinality of each label, while the **EP** technique stores the cardinality of edge patterns for each triple of labels. The **sysR** technique extends EP by also storing the number of distinct source and target vertices. The **CHAIN2** technique stores the cardinality of the CHAIN pattern with 2 edges for each combination of labels. The **SSTAR2** and **TSTAR2** techniques store the cardinality of Source STAR and Target STAR patterns with 2 edges, respectively. The **CS** technique uses Characteristic Sets for labelled source star patterns. The **WJ[X]** technique applies WanderJoin with a parameter X for the number of walks. The **BS[X]** technique uses BoundSketch with a parameter X for the number of partitions. The **ECAS** technique computes exact cardinalities for all sub-patterns on demand.

4.2 LSQB benchmark

In this section, the advisor tool will be demonstrated using the LSQB benchmark [13]. A scale factor of 0.1 is employed for the dataset, and the query workload comprises queries Q1 to Q6.

It was observed that queries Q1 and Q3 generate an excessively large number of QEPs, making it impractical to enumerate and execute all of them. Additionally, some less efficient plans for query Q2 are either too slow or require an excessive amount of memory. Consequently, only the QEPs identified by the cardinality estimation (CE) techniques were executed for queries Q1, Q2, and Q3. Furthermore, obtaining the exact cardinality of all subqueries proved to be too resource-intensive, hence the CE technique ECAS was not employed for these queries.

The summary of the results is as follows. Figure 5c shows that EP improves over SYSR in both metrics. CS and WJ100 outperform SYSR in SCT but have higher end-to-end (E2E) time. C2, PC2, CS_C2, and BS8 surpass SYSR in E2E time but require higher SCTs. SS2 and BS512 perform worse in both metrics. Most techniques exceed the E2E execution time budget. Red cells indicate the amount by which the budget was exceeded. SysR satisfies both budgets.

In Figure 5a, it is shown that PC2, C2, SS2, and CS_C2 require hours to compute their statistics, whereas BS requires minutes, and SysR, EP, and CS only seconds. WJ, which does not require any statistics, is assigned a minimal default value for inclusion in the relative comparisons of Figure 5. The absolute differences in statistics computation time are more apparent in Figure 5b.

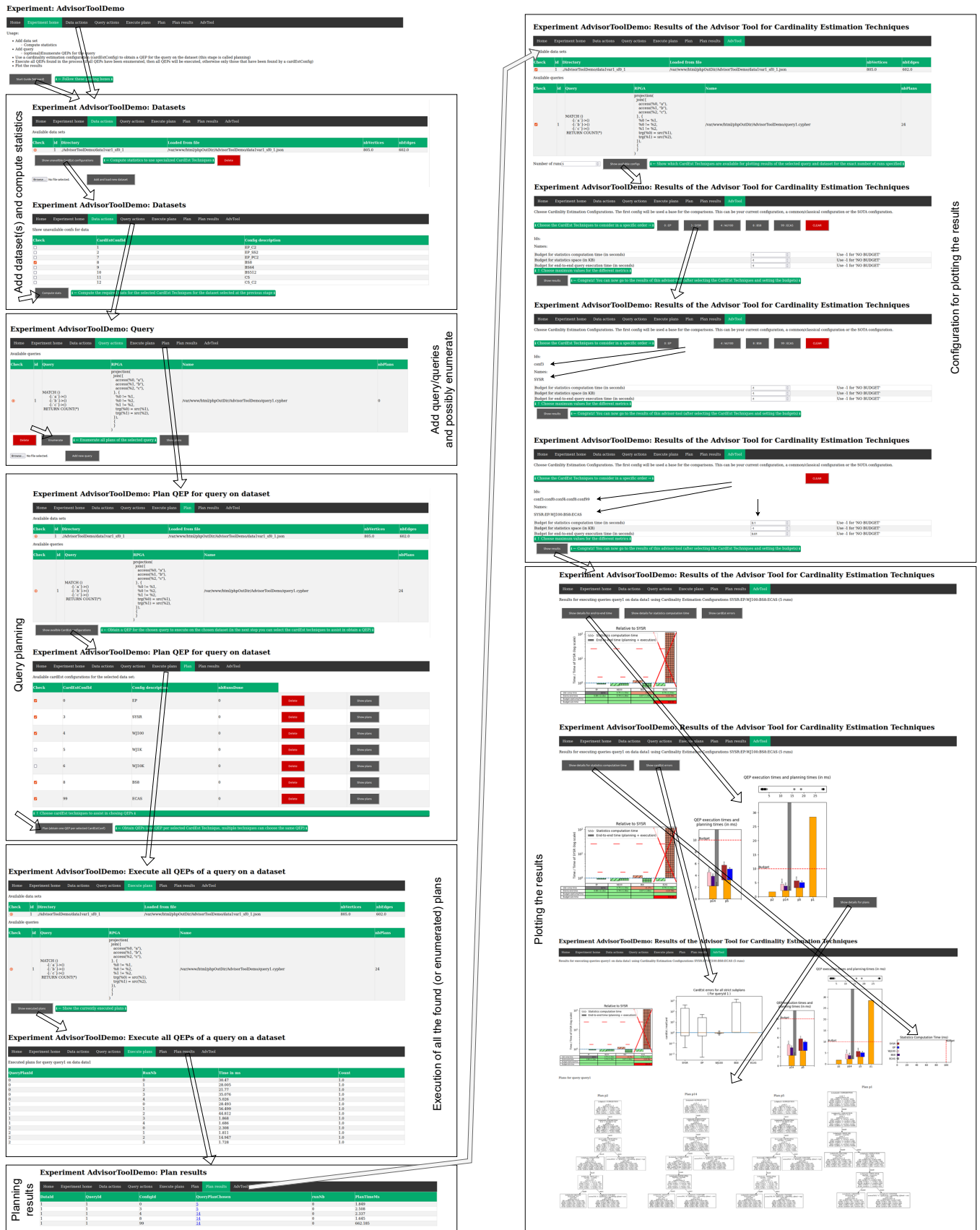


Figure 3: Screenshots of the front-end of the Advisor Tool

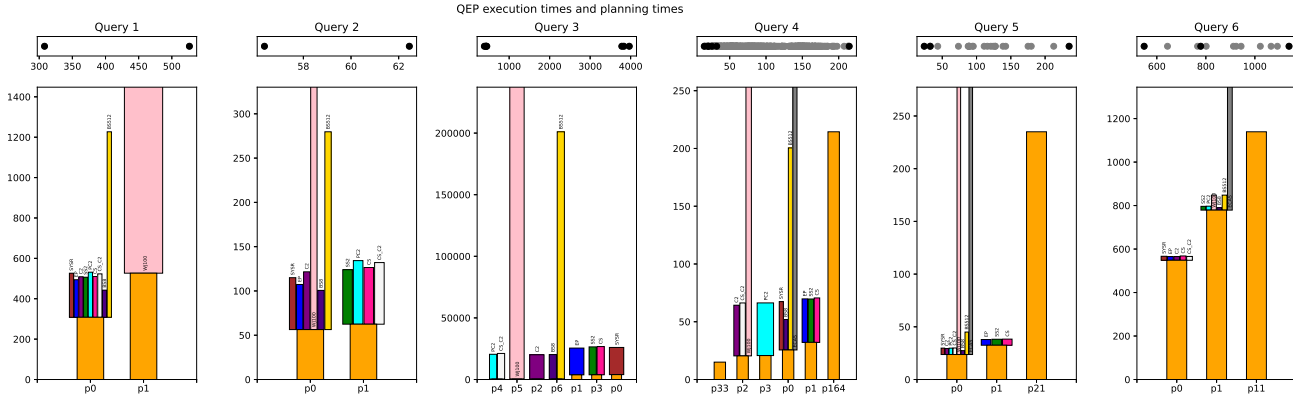


Figure 4: Details for E2E times (for LSQB sf 0.1)

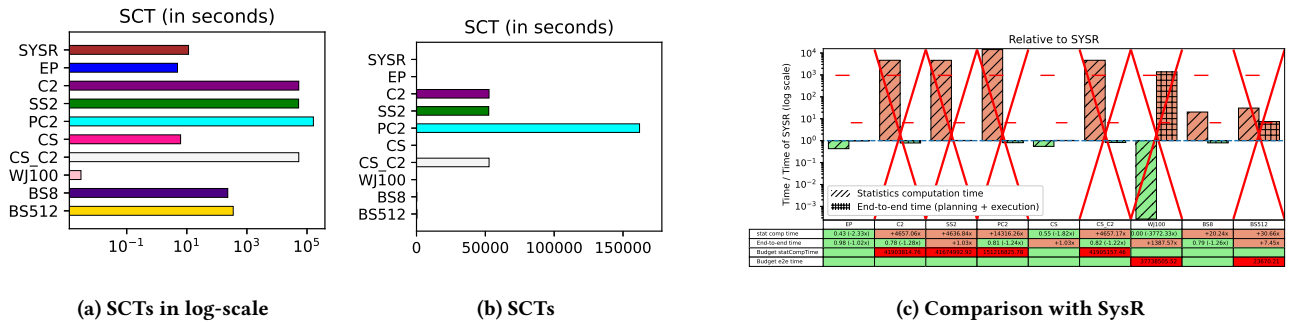


Figure 5: Statistics computation times and relative comparison with budgets (for LSQB sf 0.1)

The breakdown of the E2E time metric for each individual query is depicted in Figure 4. Techniques like WJ100, ECAS, and BS512 show exceptionally high estimate times. Queries Q3 and Q6 are the most costly to evaluate, while Q1 and Q3 are the most resource-intensive to plan. This impacts the aggregated results for queries Q1 to Q6 significantly, particularly for query Q3. Consequently, it's notable that C2, PC2, CS_C2, and BS8, which choose efficient plans for query Q3 without incurring excessive planning times, exhibit the best E2E times in Figure 5.

When imposing constraints that statistics computation times (SCTs) must not exceed three hours and the total end-to-end (E2E) execution times (summed for queries Q1 to Q6) must not exceed three minutes, only a subset of cardinality estimation (CE) techniques meet these requirements. These are SysR, EP, CS, and BS8.

Figure 5c shows that techniques such as C2, SS2, PC2, and CS_C2 surpass the SCT budget, while WJ100 and BS512 exceed the allotted E2E execution time budget. In this figure, red cells in the rows designated for budgets indicate the number of milliseconds by which the budget was exceeded. A bar that extends above the red line signifies a breach of that particular budget by the technique in question. SysR satisfies both budgets, as evidenced by the red lines being positioned above the line representing the value 1.

From these results, it is easy to conclude that EP is preferable over SysR when adhering to these budget constraints. Other techniques may be recommended if one metric is prioritized over the other. For

instance, CS could be a better choice than SysR if SCTs are deemed more critical (although, EP would be even better in this scenario), while BS8 may be favored if minimizing E2E times is the primary concern.

5 HOW MINOR CHANGES IN THE DATASET CAN BREAK YOUR ESTIMATES

The performance of cardinality estimation techniques can vary significantly across different usage scenarios. To illustrate this, we have prepared examples where minor changes in the dataset result in substantial differences in the performance of these techniques.

Consider a chain-pattern query with three edges, denoted as $q: \bigcirc \xrightarrow{a} \bigcirc \xrightarrow{b} \bigcirc \xrightarrow{c} \bigcirc$, and various dataset variants illustrated in Figure 6, parameterized by n . In all cases, executing the query $\bigcirc \xrightarrow{a} \bigcirc \xrightarrow{b} \bigcirc$ (referred to as a/b) prior to $\bigcirc \xrightarrow{b} \bigcirc \xrightarrow{c} \bigcirc$ (b/c) leads to significantly more efficient execution plans than the reverse order. Cardinality estimation techniques that do not recognize that the cardinality of a/b (denoted as $|a/b|$) is lower than that of $|b/c|$ will underperform.

We apply our advisor tool to execute the scenarios outlined above. The parameter n allows for the modification of the sizes of the generated graphs. For each variant, we generated graph with size 10^2 , 10^3 and 10^4 . The metrics are obtained by the sum over these three graphs.

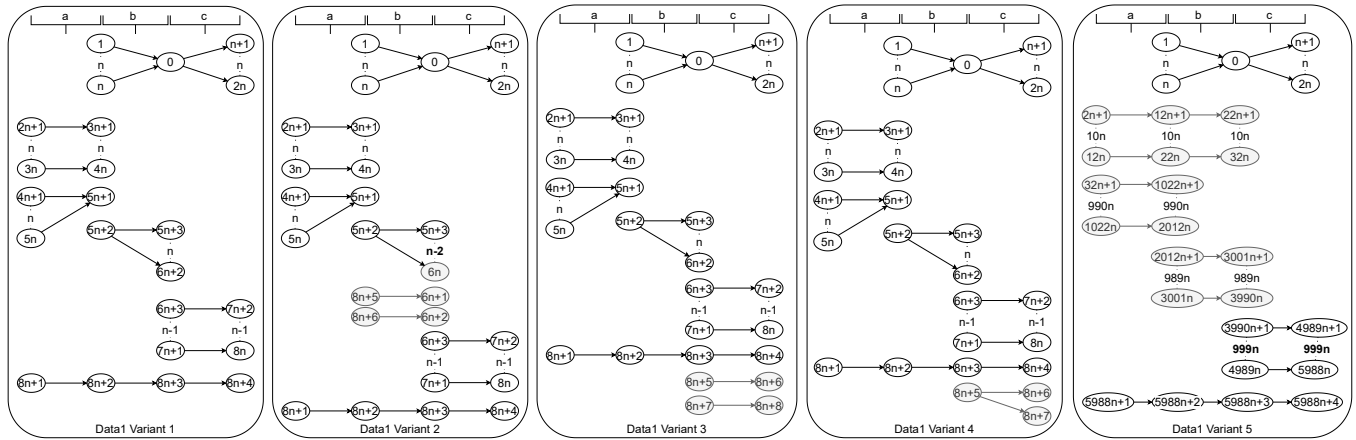


Figure 6: Data1 and variants for chain3 query

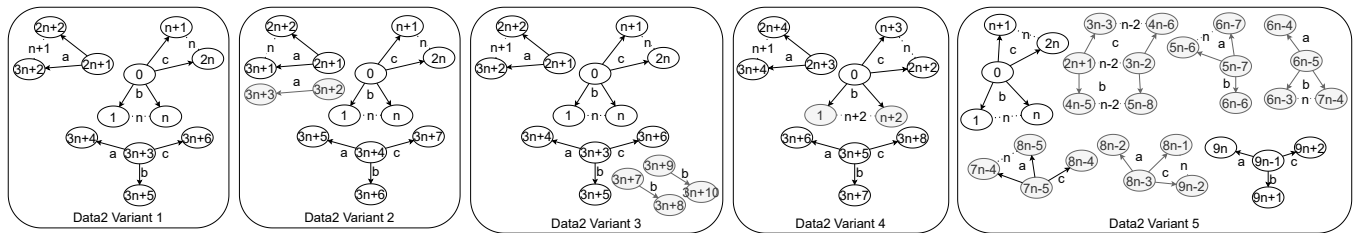


Figure 7: Data2 and variants for star3 query

For a chain-pattern query with three edges, the findings for different dataset variants are as follows (see top part of Figure 8):

- Variant 1: EP and SYSR will not select an efficient plan.
- Variant 2: EP will not select an efficient plan, but SYSR will.
- Variant 3: EP will select an efficient plan, but SYSR will not.
- Variant 4: Both EP and SYSR will select efficient plans.
- Variant 5: WJ100 is unlikely to select an efficient plan and requires a significantly longer estimation time.

For a star-pattern query with three edges, the findings for different dataset variants are as follows (see bottom part of Figure 8):

- Variant 1: EP and SYSR will not select an efficient plan.
- Variant 2: EP will not select an efficient plan, but SYSR will.
- Variant 3: EP will select an efficient plan, but SYSR will not.
- Variant 4: Both EP and SYSR will select efficient plans.
- Variant 5: CS will not select an efficient plan.

Small changes in the dataset can significantly impact the performance of CE techniques. For both the chain and the star pattern queries, the SYSR technique exhibits strong performance in Variant 2, where EP performs poorly. Thus, SYSR is highly recommended over EP in this context. Conversely, in Variant 3, the situation is reversed, with EP outperforming SYSR.

Based on these observations, we conclude that without considering the specifics of the dataset, it is challenging to provide satisfactory guidelines on the appropriate use of different cardinality estimation techniques. To further understand the influence

of specific scenarios on the performance of these techniques, our advisor tool, HOMERUN, can be used. Our approach underscores the importance of contextual analysis in the effective application of cardinality estimation methods.

6 CONCLUSION

Selecting appropriate techniques for estimating the cardinality of subqueries during query planning is crucial for optimizing performance. However, the decision-making process is complex due to the variety of available techniques, each with its own advantages and disadvantages, and the necessity to make different trade-offs in various scenarios.

To assess the performance of different cardinality estimation techniques in diverse scenarios, we have developed an advisor tool HOMERUN. This tool inputs a specific dataset and query workload, representing a particular scenario, along with a set of cardinality estimation techniques for comparison. It then produces a detailed analysis which facilitates the comparison of the performance of these estimation techniques within the given scenario.

The utility of the advisor tool has been showcased using the LSQB benchmark and a range of synthetic scenarios. These synthetic scenarios were specifically designed to illustrate the dependency of cardinality estimation technique performance on the particularities of the scenario. It was observed that minor changes in the data could lead to drastically different outcomes. Therefore, any guidelines for the selection of a particular cardinality estimation technique must consider the specifics of the



Figure 8: Relative comparison results with SYSR as base for the data variants for the chain and star queries

usage scenario. This approach ultimately underscores the need for a nuanced, context-sensitive analysis when determining the most suitable cardinality estimation method for a given situation.

Acknowledgements. This project has received funding from the European Union’s Horizon Europe framework programme under grant agreement No. 101058573.

REFERENCES

[1] The LDBC Social Network Benchmark (version 2.2.2)

[2] Aboulnaga, A., Alameldeen, A.R., Naughton, J.F.: Estimating the selectivity of xml path expressions for internet scale applications. In: VLDB. vol. 1, pp. 591–600. Citeseer (2001)

[3] Cai, W., Balazinska, M., Suciu, D.: Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. In: Proceedings of the 2019 International Conference on Management of Data. pp. 18–35 (2019)

[4] Han, Y., Wu, Z., Wu, P., Zhu, R., Yang, J., Tan, L.W., Zeng, K., Cong, G., Qin, Y., Pfadler, A., et al.: Cardinality estimation in dbms: A comprehensive benchmark evaluation. arXiv preprint arXiv:2109.05877 (2021)

[5] Hilprecht, B., Schmidt, A., Kulessa, M., Molina, A., Kersting, K., Binnig, C.: Deepdb: Learn from data, not from queries! arXiv preprint arXiv:1909.06607 (2019)

[6] Kipf, A., Kipf, T., Radke, B., Leis, V., Boncz, P., Kemper, A.: Learned cardinalities: Estimating correlated joins with deep learning. arXiv preprint arXiv:1809.06677

- (2018)
- [7] van Leeuwen, W., Fletcher, G., Yakovets, N.: A general cardinality estimation framework for subgraph matching in property graphs. *IEEE Transactions on Knowledge and Data Engineering* 35(6), 5485–5505 (2023). <https://doi.org/10.1109/TKDE.2022.3161328>
 - [8] van Leeuwen, W., Mulder, T., van de Wall, B., Fletcher, G., Yakovets, N.: Avant-graph query processing engine. *Proceedings of the VLDB Endowment* 15(12), 3698–3701 (2022)
 - [9] Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., Neumann, T.: How good are query optimizers, really? *Proceedings of the VLDB Endowment* 9(3), 204–215 (2015)
 - [10] Leis, V., Radke, B., Gubichev, A., Kemper, A., Neumann, T.: Cardinality estimation done right: Index-based join sampling. In: *Cidr* (2017)
 - [11] Leis, V., Radke, B., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., Neumann, T.: Query optimization through the looking glass, and what we found running the join order benchmark. *The VLDB Journal* 27, 643–668 (2018)
 - [12] Li, F., Wu, B., Yi, K., Zhao, Z.: Wander join: Online aggregation via random walks. In: *Proceedings of the 2016 International Conference on Management of Data*. pp. 615–629 (2016)
 - [13] Mhedhbi, A., Lissandrini, M., Kuiper, L., Waudby, J., Szárnyas, G.: Lsqb: a large-scale subgraph query benchmark. In: *Proceedings of the 4th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*. pp. 1–11 (2021)
 - [14] Neumann, T., Moerkotte, G.: Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins. In: *2011 IEEE 27th International Conference on Data Engineering*. pp. 984–994. IEEE (2011)
 - [15] Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. *ACM Sigmod Record* 25(2), 294–305 (1996)
 - [16] Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*. pp. 23–34 (1979)
 - [17] Stefanoni, G., Motik, B., Kostylev, E.V.: Estimating the cardinality of conjunctive queries over rdf data using graph summarisation. In: *Proceedings of the 2018 World Wide Web Conference*. pp. 1043–1052 (2018)
 - [18] Wu, W.: Sampling-based cardinality estimation algorithms: A survey and an empirical evaluation
 - [19] Wu, Z., Shaikhha, A., Zhu, R., Zeng, K., Han, Y., Zhou, J.: Bayescard: Revitalizing bayesian frameworks for cardinality estimation. *arXiv preprint arXiv:2012.14743* (2020)
 - [20] Yang, Z., Kamsetty, A., Luan, S., Liang, E., Duan, Y., Chen, X., Stoica, I.: Neurocard: one cardinality estimator for all tables. *arXiv preprint arXiv:2006.08109* (2020)