Machine learning for query selectivity estimation in relational databases

PhD defence

Max Halford[1,2]
Philippe Saint-Pierre[1] Franck Morvan[2]

[1] Toulouse Institute of Mathematics (IMT)

[2] Toulouse Institute of Informatics Research (IRIT)

October 12, 2020

Outline

Motivation

A relation

Name	Nationality	Hair	Shop	Item
Shinzo	Japanese	Black	7-Eleven	Ramen
Shinzo	Japanese	Black	7-Eleven	Crisps
Stefan	Swedish	Blond	Ikea	Meatballs
Stefan	Swedish	Blond	Ikea	Chair
Stefan	Swedish	Blond	Ikea	Meatballs
Stefan	Swedish	Blond	7-Eleven	Crisps
Donald	American	Blond	Walmart	Milkshake
Donald	American	Blond	Walmart	Crisps

Normalisation

Person	Shop	Item
P1	S1	I1
P1	S1	I2
P2	S2	I3
P2	S2	I4
P2	S2	I3
P2	S1	I2
P3	S3	I5
P3	S3	I2

ID	Name
S1	7-Eleven
S2	Ikea
S3	Walmart

ID	Name
I1	Ramen
I2	Crisps
I3	Meatballs
I4	Chair
I5	Milkshake

ID	Name	Nationality	Hair
P1	Shinzo	Japanese	Black
P2	Stefan	Swedish	Blond
P3	Donald	American	Blond

Queries

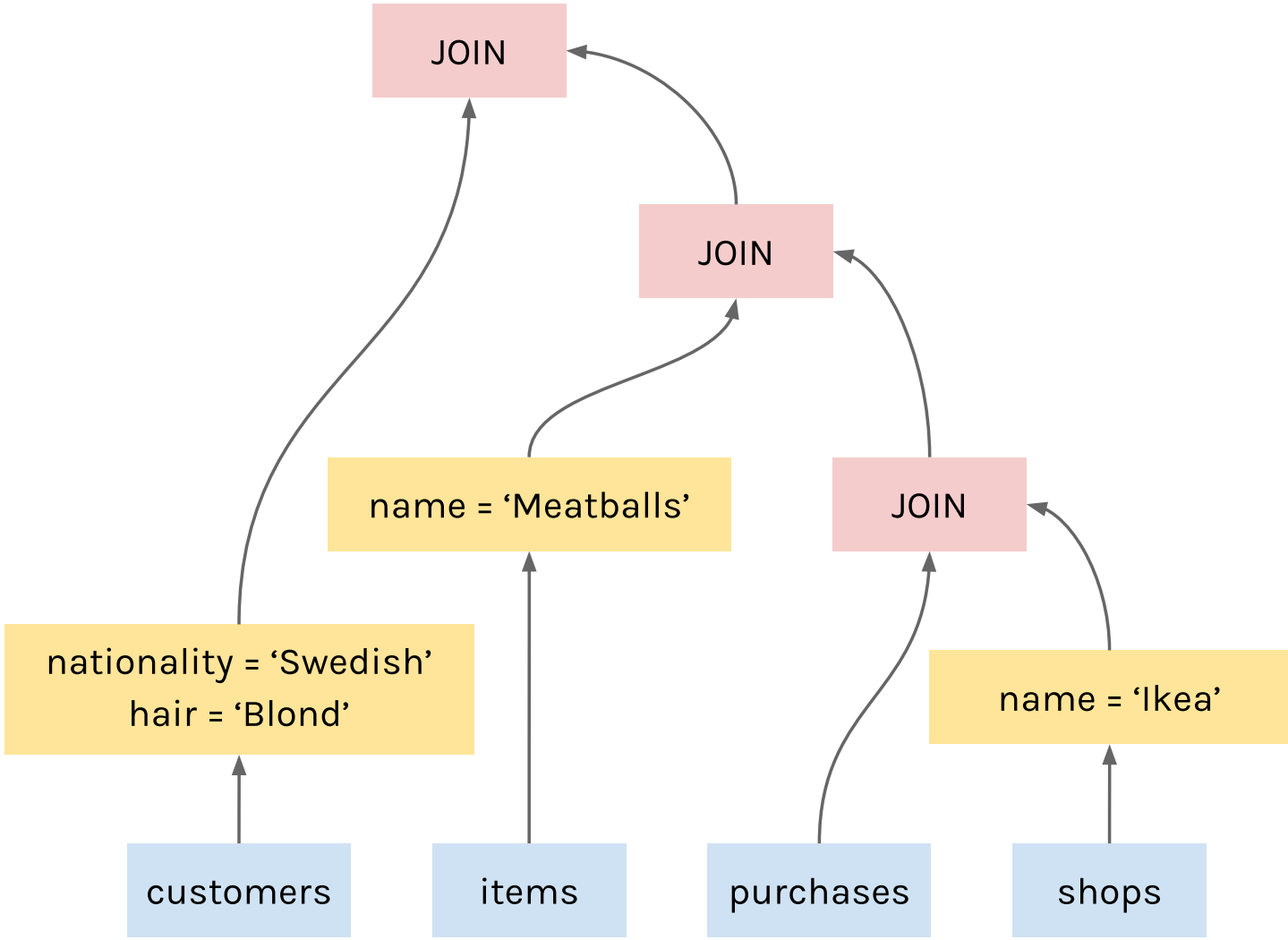
How many blond Swedes bought meatballs from Ikea?

```
SELECT COUNT(*)

-- Relations
FROM customers, shops, items, purchases

-- Joins
WHERE purchases.customer = customers.id
AND purchases.shop = shops.id
AND purchases.item = items.id

-- Filters
AND customers.nationality = 'Swedish'
AND customers.hair = 'Blond'
AND items.name = 'Meatballs'
AND shops.name = 'Ikea'
```

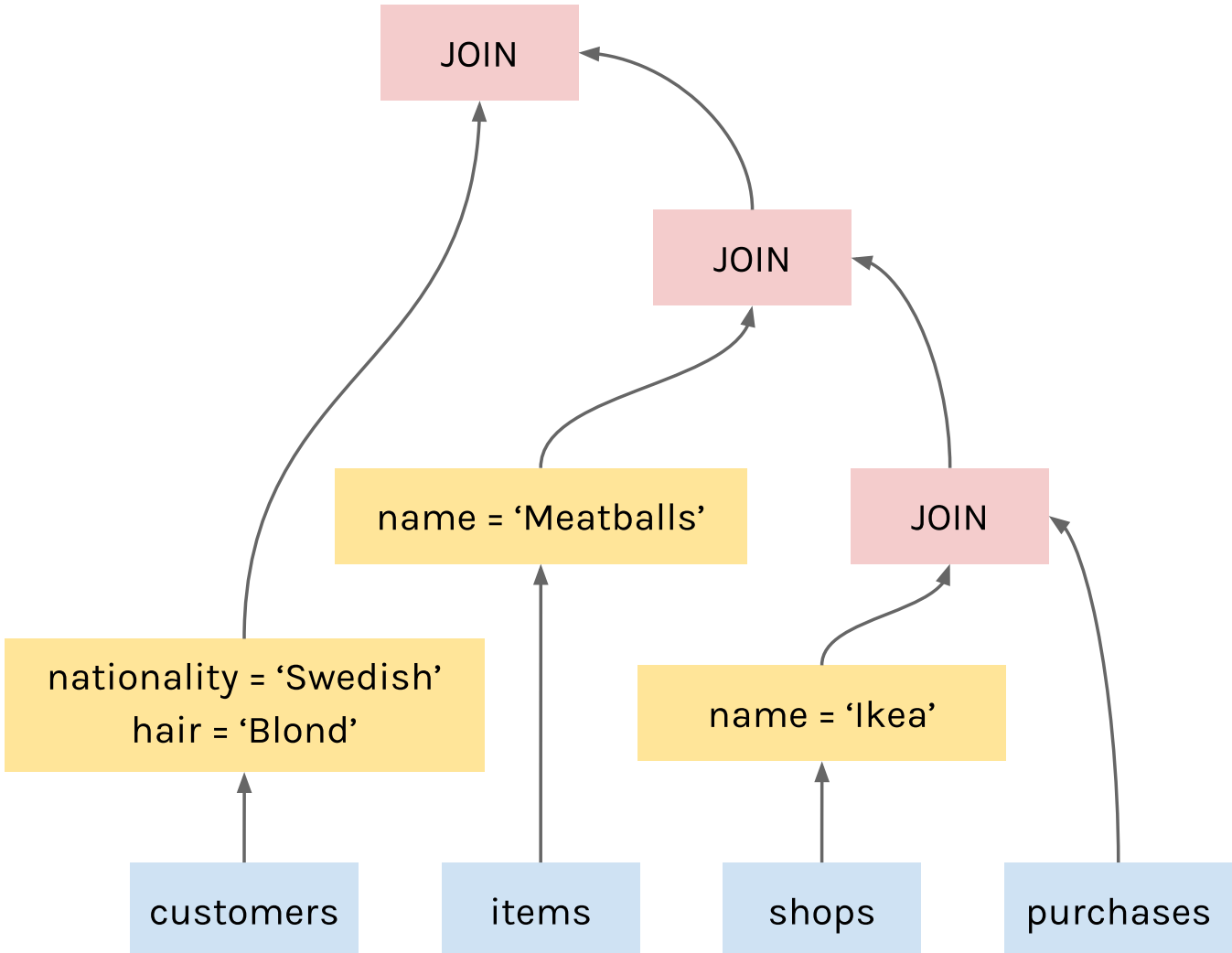
Query optimisation

1. A user issues an SQL query
2. Each query can be answered with many different **query execution plans**
3. The **query optimiser** searches for the fastest execution plan
4. The best plan is executed, the results are returned to the user
5. **Difficulty**: the exact cost of each plan is unknown before executing it

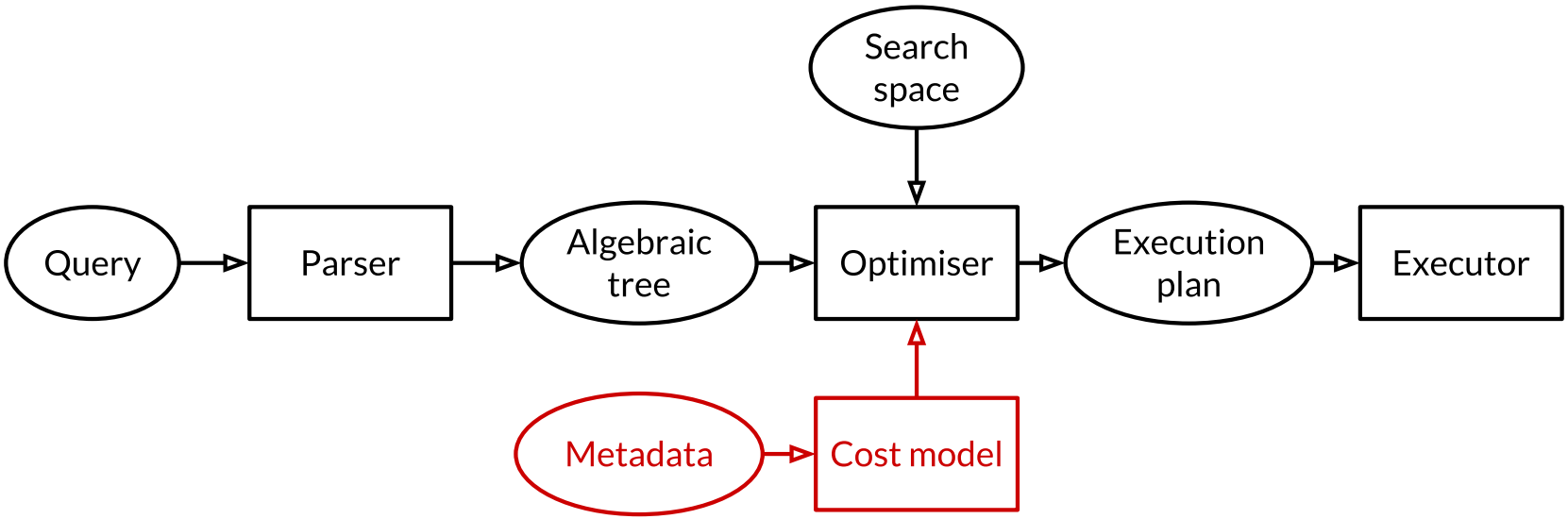
Join order matters (1)



Join order matters (2)



Cost-based query optimisation



Query optimisation time is part of the query response time!

Cost model

- Cost of operator mostly depends on number of tuples to process, called the **selectivity**
- Selectivity is extremely difficult to estimate [WCZ+13]
- Errors propagate exponentially [IC91]
- Seminal (simple) ideas from [SAC+79] are still very much in use [1]

[1] `https://www.postgresql.org/docs/13/planner-stats.html`

Selectivity estimation

- How many customers?
- How many Swedish customers?
- How many purchases from Ikea?
- How many purchases from Ikea for meatballs?
- How many Swedish bought meatballs from Ikea?

The goal of this PhD is to propose efficient methods that can answer these kind of queries.

Selectivity estimation is difficult because of dependencies, even more so when they scattered across relations.

Assumptions

1. **Attribute value independence** (AVI): attributes are independent with each other
2. **Join uniformity**: attribute distributions do not change after joins
3. **Join predicate independence**: join selectivities can be computed independently

In practice, all these assumptions go out the window.
The goal is to relax them.

State of the art

Dummy strategies

1. Scan the data on the fly: takes too long
2. Memorise all possible combinations: takes too much space

A compromise between time and space has to be made

Existing approaches

1. Sampling
2. Supervised learning
3. Unsupervised learning

Sampling

- Idea: estimate a selectivity by running a query on a sample of the relations
- The sample can be constructed:
  1. online: expensive because of on the fly disk access, but accurate
  2. offline: can be done during downtime, but needs to be refreshed
- Different methods:
  - Single relation [PSC84, LN90] (works well but has a high inference cost)
  - Multiple relations [Olk93, VMZC15, ZCL+18] (empty-join problem)

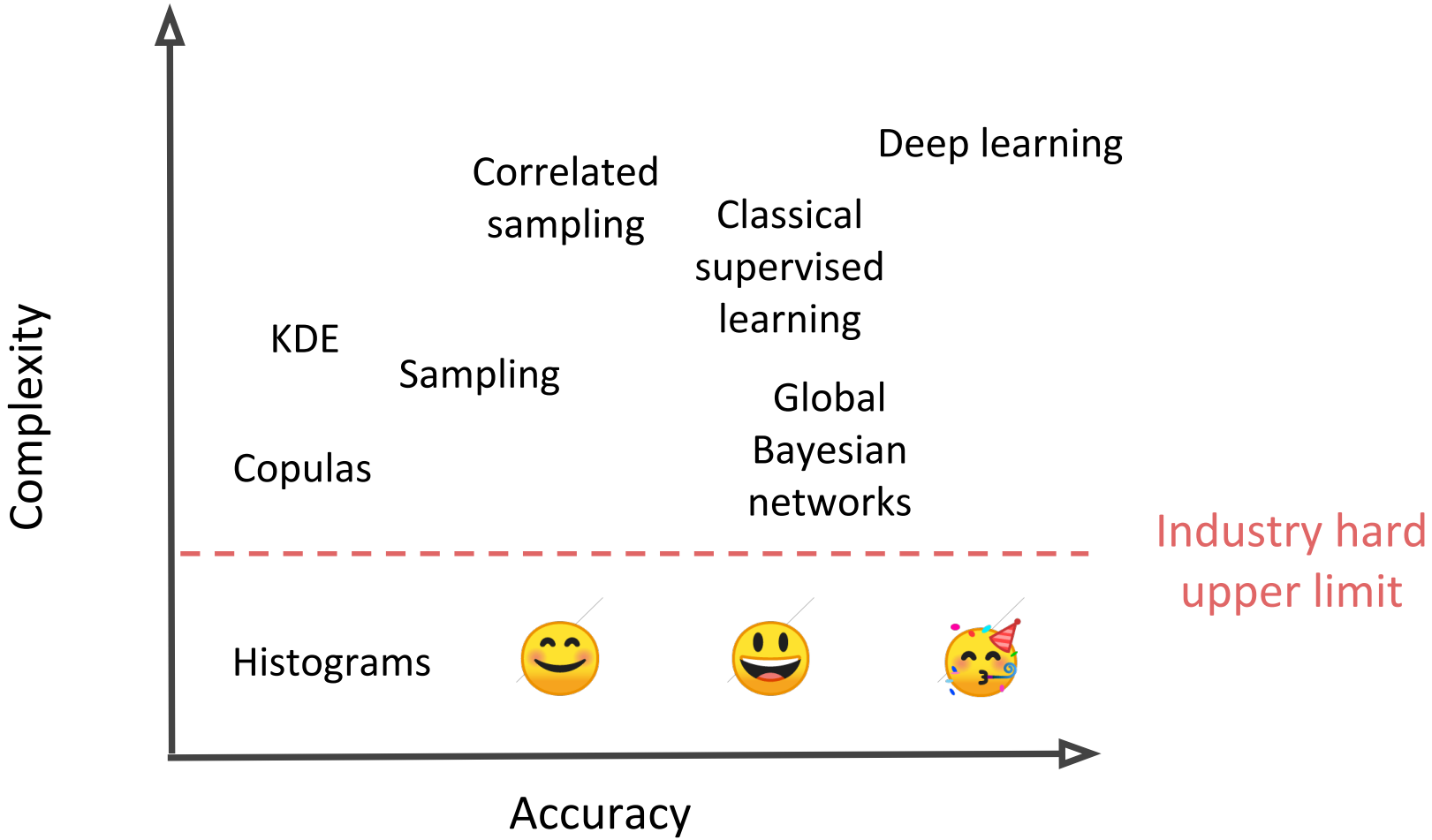
Supervised learning

- Idea: frame selectivity estimation as a supervised learning problem:
  1. Features: queries issued by users
  2. Targets: the selectivity of each query
  3. Goal: minimise some metric
- Different methods:
  - Query feedback memorisation [SLMK01] (useless for unseen queries)
  - "Classical" machine learning [CR94, AÇR+12, LXY+15]
  - Deep learning [KKR+18, DWN+19, WXQ+20]
- Little to no work takes into account **concept drift** and **cold start** issues

Unsupervised learning

- Idea: build a statistical synopsis of the database to perform density estimation
- The goal is to capture as much information as possible
- Domain closure: the data we have is not part of some larger sample
- Different methods:
  - Unidimensional [IC93, TCS13, HKM15] (textbook approach)
  - Multidimensional [GKTD05, Aug17] (exponential number of combinations)
  - Bayesian networks [GTK01, TDJ11] (complex compilation procedure and high inference cost)

The accuracy vs. complexity trade-off



Selectivity estimation with Bayesian networks

A statistical point of view

- A relation is made of p attributes $X_1, \ldots, X_p$
- Each attribute X_i follows an unknown distribution $P(X_i)$
- $P(X_i = x)$ gives us the probability of a predicate (e.g. `name = 'Ikea'`)
- $P(X_i)$ can be estimated, for example with a histogram
- The distribution $P(X_i, X_j)$ captures interactions between X_i and X_j (e.g. `name = 'Ikea' AND nationality = 'Swedish'`)
- Memorising $P(X_1, \ldots, X_p)$ takes $\prod_0^p |X_i|$ units of space

Independence

- Assume $X_1, \ldots, X_p$ are independent with each other
- We thus have $P(X_1, \ldots, X_p) = \prod_0^p P(X_i)$
- Memorising $P(X_1, \ldots, X_p)$ now takes $\sum_0^p |X_i|$ units of space
- We've compromised between accuracy and space
- In query optimisation this is called the **attribute value independence** (AVI) assumption

Conditional independence

- Bayes' theorem: $P(A, B) = P(B \mid A) \times P(A)$
- Example: $P(hair, country) = P(hair \mid country)P(country)$
- A are B are **conditionally independent** if C determines them
- In that case $P(A, B, C) = P(A \mid C)P(B \mid C)P(C)$
- $|P(A \mid C)| + |P(B \mid C)| + |P(C)| < |P(A, B, C)|$
- Conditional independence can save space without compromising on accuracy!

Example

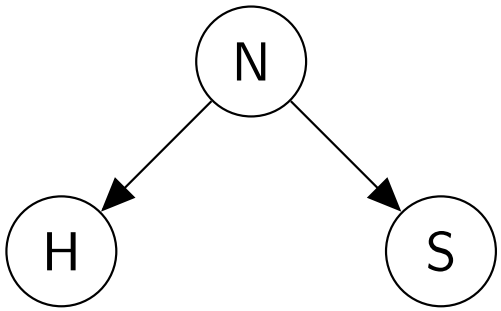
nationality	hair	salary
Swedish	Blond	42000
Swedish	Blond	38000
Swedish	Blond	43000
Swedish	Brown	37000
American	Brown	35000
American	Brown	32000

- Truth: $P(Swedish, Blond) = \frac{3}{6} = 0.5$
- With independence:
  - $P(Swedish) = \frac{4}{6}$
  - $P(Blond) = \frac{3}{6}$
  - $P(Swedish, Blond) \simeq P(Swedish) \times P(Blond) = \frac{2}{6} = 0.333$
- With conditional independence:
  - $P(Blond \mid Swedish) = \frac{3}{4}$
  - $P(Swedish, Blond) = P(Blond \mid Swedish) \times P(Swedish) = \frac{3 \times 4}{4 \times 6} = 0.5$

Bayesian networks

- Assuming full independence isn't accurate enough
- Memorising all value combinations takes too much space
- Pragmatism: **some variables are independent, some aren't**
- Bayes' theorem + pragmatism = Bayesian networks
- Conditional independences are organised in a graph
- Each node is a variable and is dependent with its parents

Example



American	Swedish
0.333	0.666

Table: $P(nationality)$

	Blond	Brown
American	0	1
Swedish	0.75	0.25

Table: $P(hair \mid nationality)$

	< 40k	> 40k
American	1	0
Swedish	0.5	0.5

Table: $P(salary \mid nationality)$

Structure learning

- In a tree, each node has 1 parent, benefits:
  - 2D conditional distributions (1D for the root)
  - Low memory footprint
  - Low inference time
- Use of Chow-Liu trees [CL68]
  1. Compute mutual information (MI) between each pair of attributes
  2. Let the MI values define fully connected graph G
  3. Find the maximum spanning tree (MST) of G
  4. Orient the MST (i.e. pick a root) to obtain a directed graph

Selectivity estimation

- Use of **variable elimination** [CDLS06]
- Works in $\mathcal{O}(n)$ time for trees [RS86]
- **Steiner tree** [HRW92] extraction to speed up the process

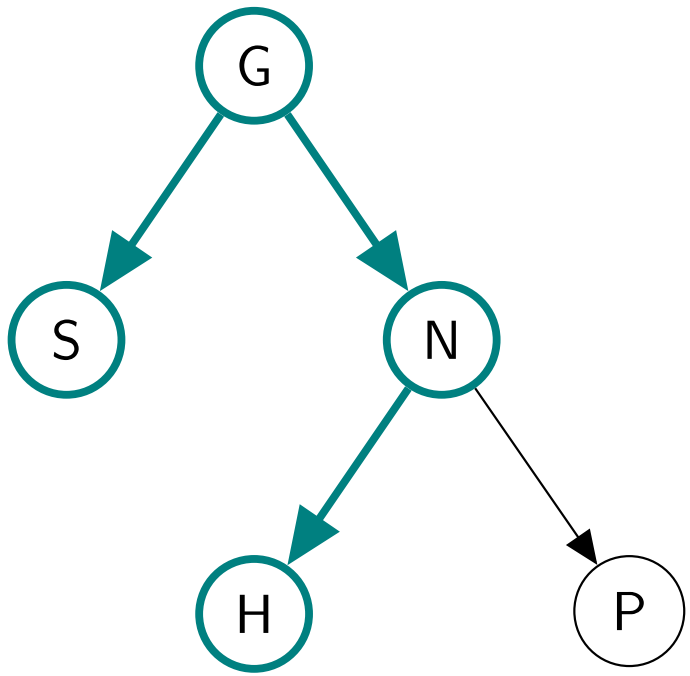


Figure: Highlighted Steiner tree containing nodes G, S, N, and H needed to compute H's marginal distribution

The stepping stone

- We managed to soften the AVI assumption within a relation [HSPM19]
- The next step is to take into account dependencies across different relations [HSPM20]

Bad assumptions

- Up to now, we assumed $P_C(Swedish) = P_{C \bowtie P}(Swedish)$ (join uniformity assumption)
- We also assumed $P_{C \bowtie P \bowtie S}(Swedish, Ikea) = P_C(Swedish) \times P_S(Ikea)$ (AVI assumption)
- Both of these assumptions are wrong in practice

Dependency preservation assumption

- Example factorisation: $P_C(A, B) = P_C(A \mid B)P_C(B)$
- Assumption: attribute dependencies are preserved after joins
- Example: $P_{C \bowtie P}(A, B) = P_C(A \mid B)P_{C \bowtie P}(B)$
- We don't need to know $P_{C \bowtie P}(A \mid B)$!

Illustration

- I know how many customers are Swedish and blond
- I know how many Swedes made purchases
- I assume that all Swedes are as likely to make purchases
- Consequently, I know many blond Swedes made purchases

This is much softer than the join uniformity assumption

Example *without* the assumption

Shop	Customer
1	1
2	1
3	1
4	1
5	2
6	3
7	5

Table: Purchases

Customer	Nationality	Hair
1	Swedish	Blond
2	Swedish	Blond
3	Swedish	Brown
4	American	Blond
5	American	Brown

Table: Customers

- $P_{C \bowtie P}(Blond, Swedish) = \frac{5}{7}$
- $P_C(Swedish) = \frac{3}{5}$
- $P_C(Blond \mid Swedish) = \frac{2}{3}$
- $\hat{P}_{C \bowtie P}(Blond, Swedish) \simeq \frac{2}{3} \times \frac{3}{5} = \frac{2}{5}$ (44% underestimate)

Example *with* the assumption

Shop	Customer
1	1
2	1
3	1
4	1
5	2
6	3
7	5

Table: Purchases

Customer	Nationality	Hair
1	Swedish	Blond
2	Swedish	Blond
3	Swedish	Brown
4	American	Blond
5	American	Brown

Table: Customers

- $P_{C \bowtie P}(Blond, Swedish) = \frac{5}{7}$
- $P_{C \bowtie P}(Swedish) = \frac{6}{7}$
- $P_{C}(Blond \mid Swedish) = \frac{2}{3}$
- $\hat{P}_{C \bowtie P}(Blond, Swedish) \simeq \frac{2}{3} \times \frac{6}{7} = \frac{4}{7}$ (20% underestimate)

Independent Bayesian networks

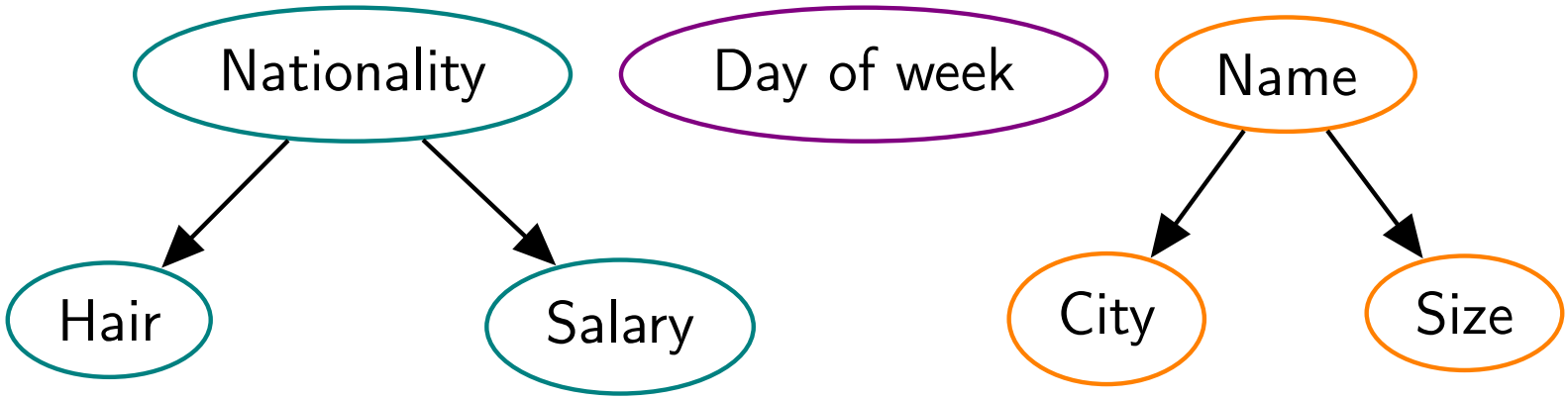


Figure: Separate Bayesian networks of customers, shops, and purchases

Linked Bayesian network

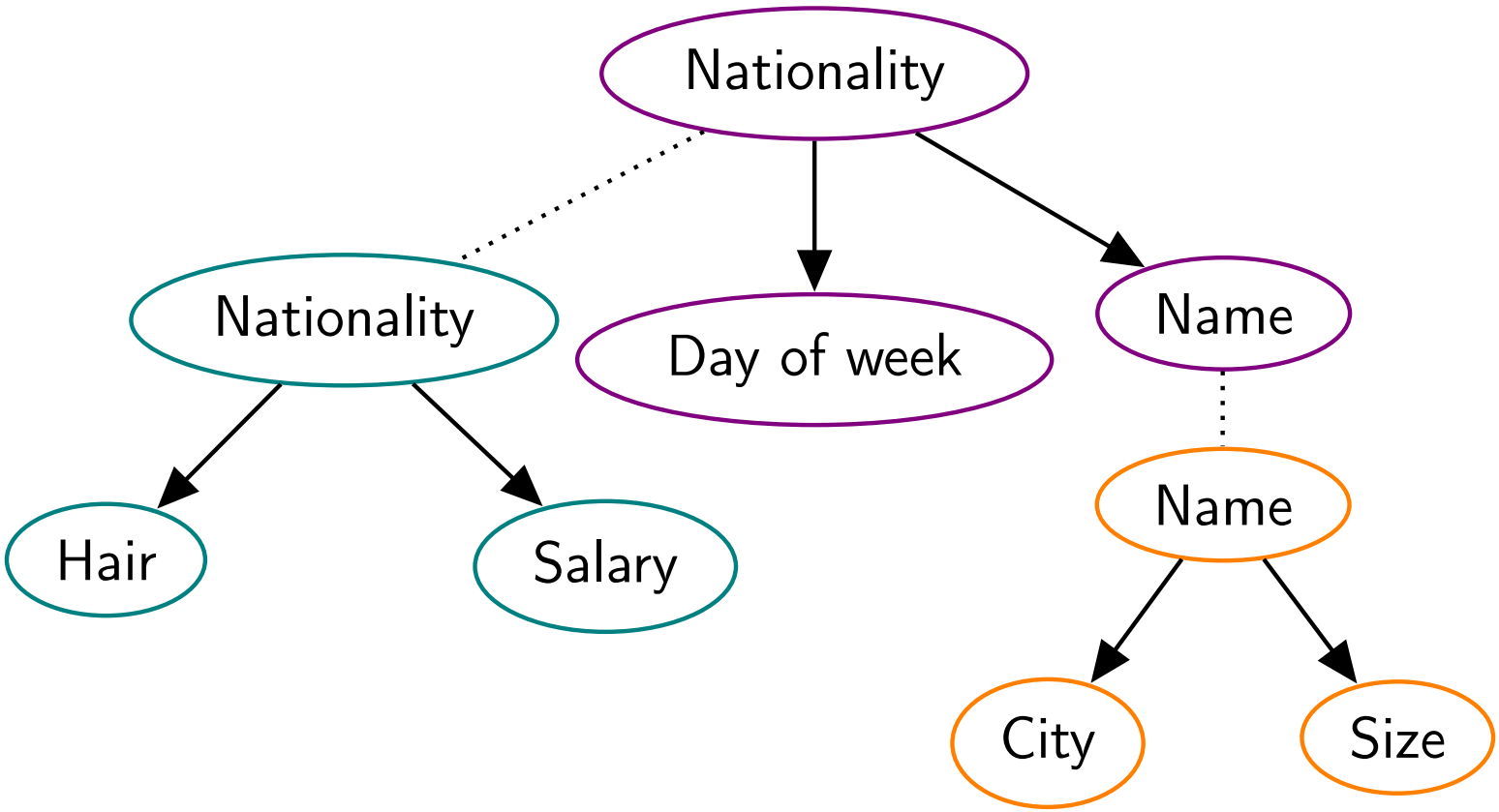


Figure: Linked Bayesian network of customers, shops, and purchases

"Unrolled" linked Bayesian network

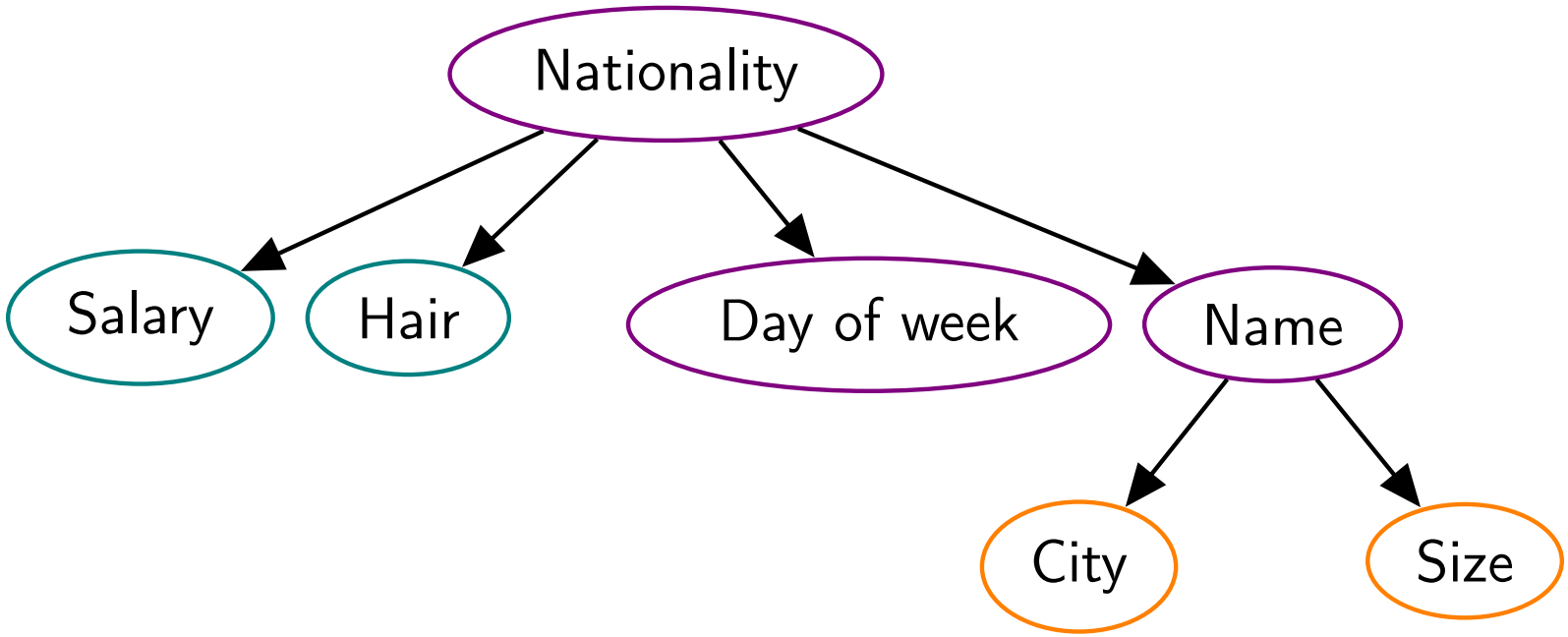


Figure: Unrolled version of the linked Bayesian network of customers, shops, and purchases

Including more attributes

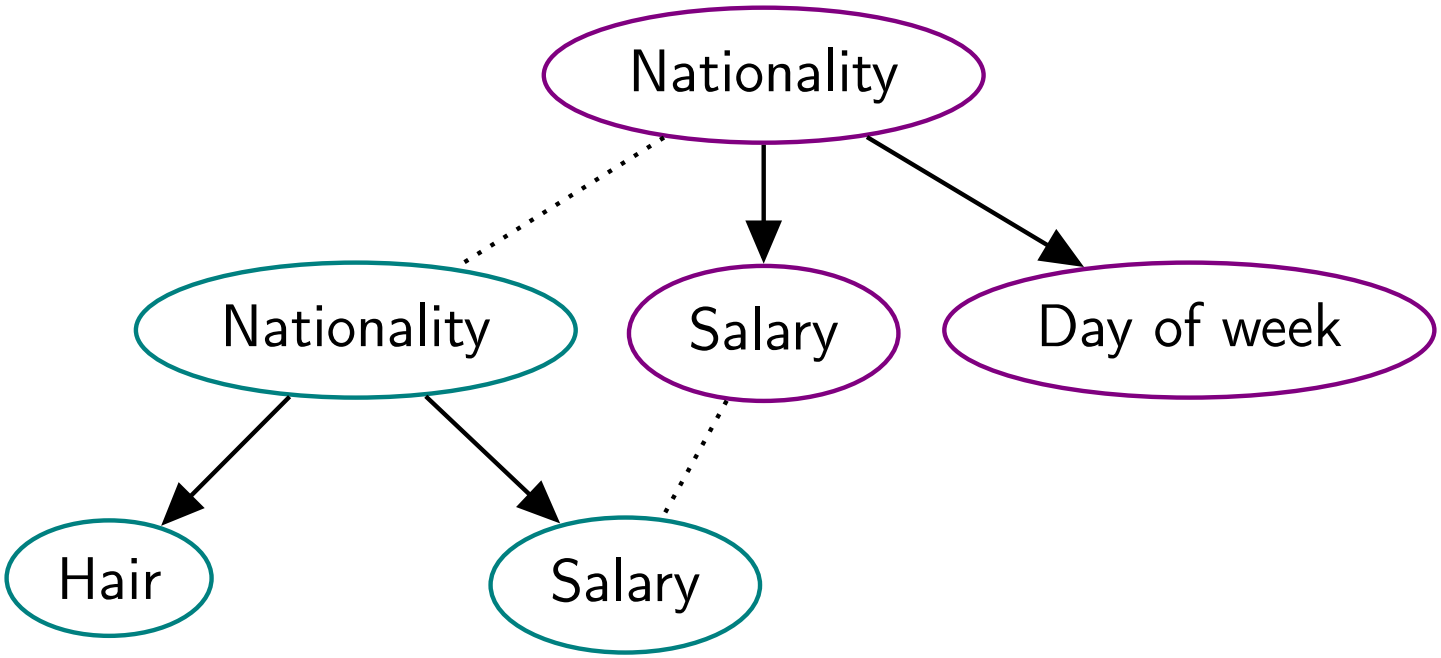


Figure: Linked Bayesian network of customers and purchases

Benefits of linked Bayesian networks

1. Only one attribute of each relation has to be included
2. New distributions are obtained for free because of the attribute dependency preservation assumption
3. By trying to relax the join uniformity assumption, we also relaxed the attribute value independence assumption across relations!
4. The same inference algorithm used for a single Bayesian network can be used on the unrolled version of a linked Bayesian network

Experimental setup – environment

- Measure accuracy using q-error [LRG+18]:

$$q(y, \hat{y}) = \frac{max(y, \hat{y})}{min(y, \hat{y})}$$

- Measure inference speed
- Measure compilation time and storage requirements
- Use 5,122,790 queries derived from the *Join Order Benchmark* [LGM+15]

Experimental setup – compared methods

We benchmarked the following methods:

1. PostgreSQL
2. Random sampling [OR86]
3. Correlated sampling [VMZC15]
4. MSCN (deep learning) [KVM+19]
5. Global Bayesian network [TDJ11]
6. Independent Bayesian networks [HSPM19] (us)
7. Linked Bayesian with $k = 1$ (us)
8. Linked Bayesian with $k = 2$ (us)

Figure: Sorted q-errors for all queries by method

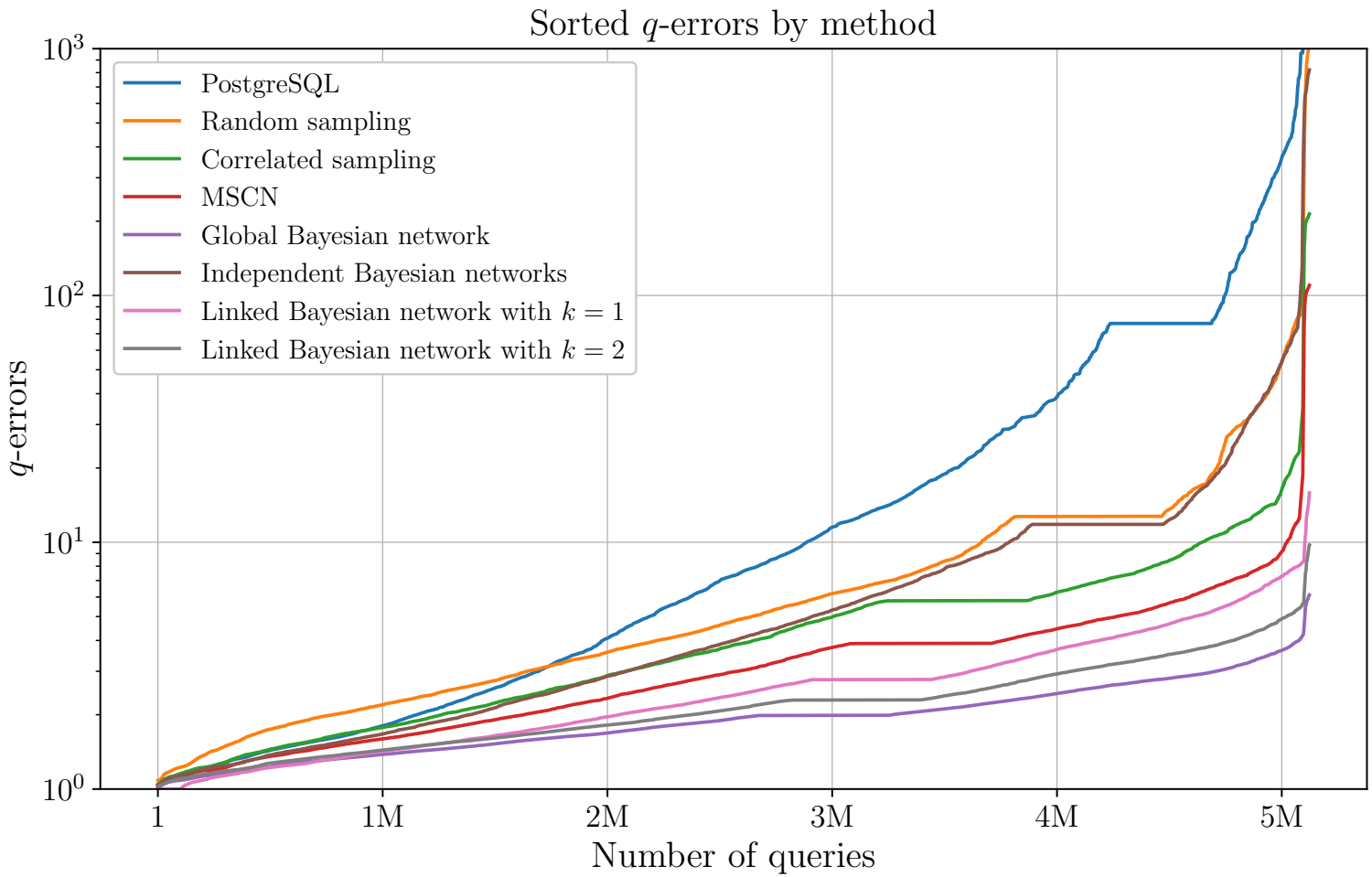


Table: q-error statistics for each method

	median	95th	99th	max	average
PostgreSQL	7.32	185.84	707.21	10906.17	77.01
Sampling	4.79	33.17	81.34	1018.43	12.71
Correlated sampling	3.83	12.63	22.72	214.1	5.79
MSCN	2.99	7.47	12.49	110.56	3.89
Global BN	1.95	3.22	4.01	7.45	1.99
Independent BN	4.0	32.9	76.91	820.46	11.82
Linked BN $k = 1$	2.41	6.15	8.07	21.09	2.79
Linked BN $k = 2$	2.13	4.26	5.23	12.6	2.3

Table: Average inference time in milliseconds for each method with respect to the number of joins

	No joins	1 join	2 to 5 joins	6 joins or more
PostgreSQL	2.3	2.6	3.6	8.4
Sampling	19.6	36.2	120.2	268.4
Correlated sampling	20.4	155.7	280.6	493.4
MSCN	135.9	312.2	343.3	387.6
Global BN	84.3	116.1	145.8	236.1
Independent BN	8.3	10.9	12.6	12.1
Linked BN $k = 1$	9.5	12.8	14.1	15.2
Linked BN $k = 2$	10.1	12.9	14.3	16.4

Table: Computational requirements of the compilation phase per method

	Construction time	Storage size
PostgreSQL	5s	12KB
Sampling	7s	276MB
Correlated sampling	32s	293MB
MSCN	15m8s	37MB
Global BN	24m45s	429KB
Independent BN	55s	217KB
Linked BN $k = 1$	2m3s	322KB
Linked BN $k = 2$	2m8s	464KB

Correcting selectivities with machine learning

Recap

- Bayesian networks are a good candidate to replace histograms
- However, a lot of query optimisers are reluctant to modify their cost model
- Can we improve an existing cost model by learning from its mistakes?

Idea

- Assume we have a cost model
- The true costs are denoted y_i
- The estimated costs are denoted $\hat{y}_i$
- The average relative error is $e = \frac{1}{n} \sum_{i=1}^{n} \frac{\hat{y}_i}{y_i}$
- If $e > 1$, the model is overestimating
- If $e < 1$, the model is underestimating
- **We can divide each $\hat{y}_i$ by e to correct the model**
- Naturally, we don't know e before a workload occurs, but we can estimate it online
- Used in Microsoft SQL server [LKN+16]

Baby steps

- Assumption: the model's error depends on the complexity of the query
- Example: relative errors might be correlated with the number of joins
- We can calculate the errors e_k for each number of joins k
- When an estimate for a query is made, correct the estimate by dividing the estimate by the appropriate e_k

A statistical learning perspective

- We can take this further, and correlate the error with *features*, denoted x_i, derived from each query
- Goal: learn to predict the error from x_i
- Approach: use error feedback to update a statistical
- In other words, we can use machine learning to (try to) correlate query properties with relative errors

Supervised learning in a nutshell

- Supervised learning is a subset of machine learning
- Typically, we have a $n \times p$ matrix of n observations and p features, denoted X
- We also have a vector of n target values, denoted y
- The goal is to learn a model m which maps X to f
- Ideally, m should **generalise** and not memorise

Recent endeavours

- Many recent proposals:
  - Gradient boosting [DWNC]
  - Random forests [IB17]
  - Deep learning [KKR+18]
  - More deep learning [WJA+18]
  - Even more deep learning [WHT+19]
  - Traditional linear models [DWN+19]
- All of these proposals use a **batch** approach
- Batch models need to be retrained when new data arrives
- Queries are, by nature, endlessly streaming in
- Data distribution and queries typologies change through time (**concept drift**)
- Batch models are therefore sub-optimal
- Also, batch models are too cumbersome [HSK+19]

The benefits of online machine learning

- An online model is able to learn with one sample at a time
- Online learning is a better fit than batch learning for the purpose of selectivity estimation:
  1. An online paradigm doesn't require to store historical data
  2. An online model is always up-to-date
  3. An online model can cope with concept drift

Target encoding

- The idea of LEO [SLMK01] is to memorise past selectivities of sub-QEPs
- We can do the same in spirit, by memorising the error with respect to:
  1. relations
  2. attributes
  3. joins
  4. attribute values
  5. number of operators
- In data science, this is called **target encoding**
- Might overfit when groups are too small

Factorisation machines (FM)

FMs supersede linear models:

$$\hat{y} = w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} \langle v_i, v_j \rangle x_i x_j \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the dot product:

$$\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} \times v_{j,f} \tag{2}$$

This model parameter are thus:

$$w_0 \in \mathbb{R} \qquad w_i \in \mathbb{R}^p \qquad w_{if} \in \mathbb{R}^{p \times k} \tag{3}$$

Experimental results

- We simulate a stream of queries
- Soft and hard drift are inserted to check for robustness
- We benchmark batch models:
  - Linear regression
  - LightGBM [KMF+17]
  - MSCN [KVM+19]
  - PostgreSQL's cost model
- As well as online models:
  - Linear regression
  - Bayesian linear regression
  - Multi-layer perceptron
  - Factorization machines [Ren10]
  - Hoeffding trees [DH00]

Figure: q-errors for each method with **hard** concept drift

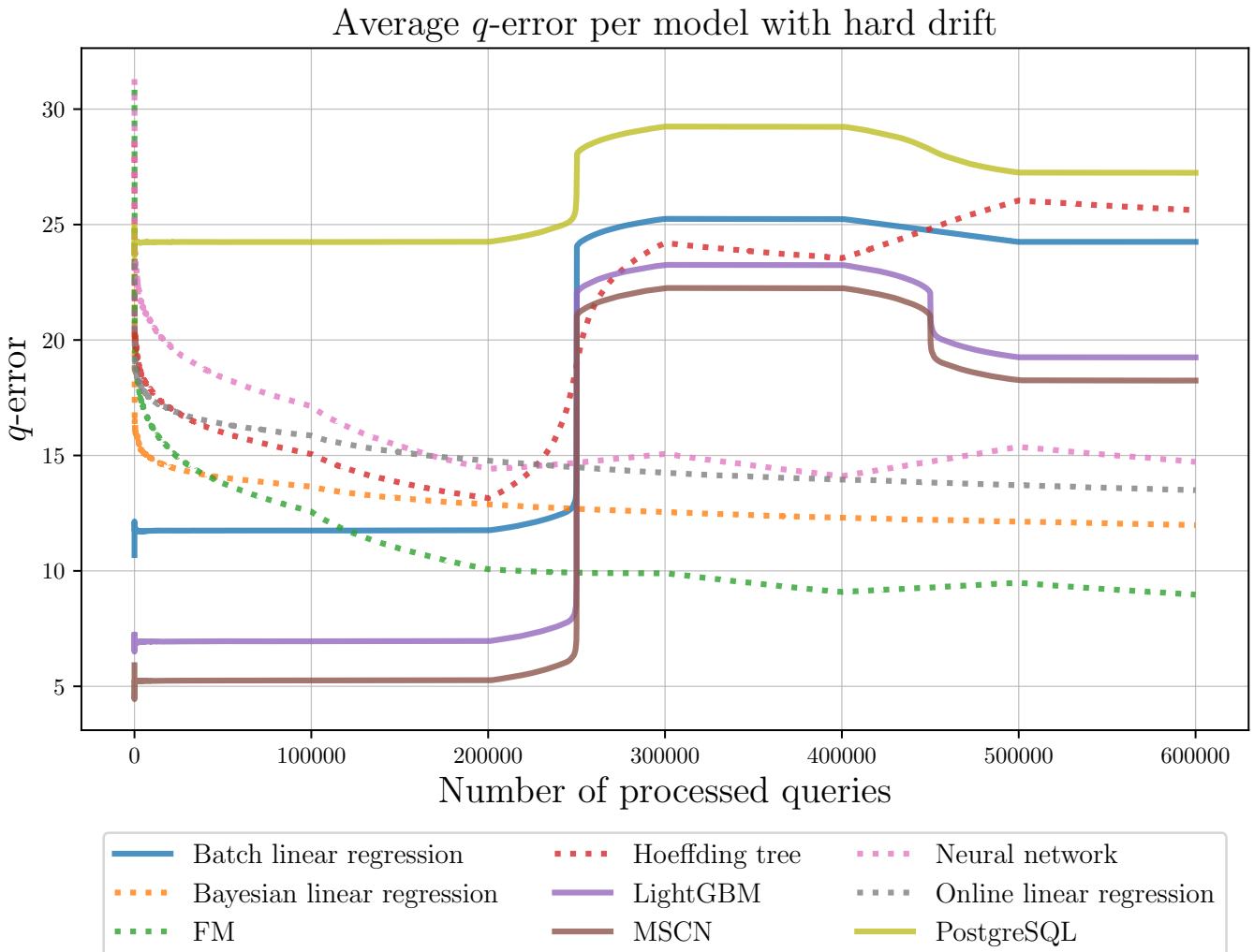
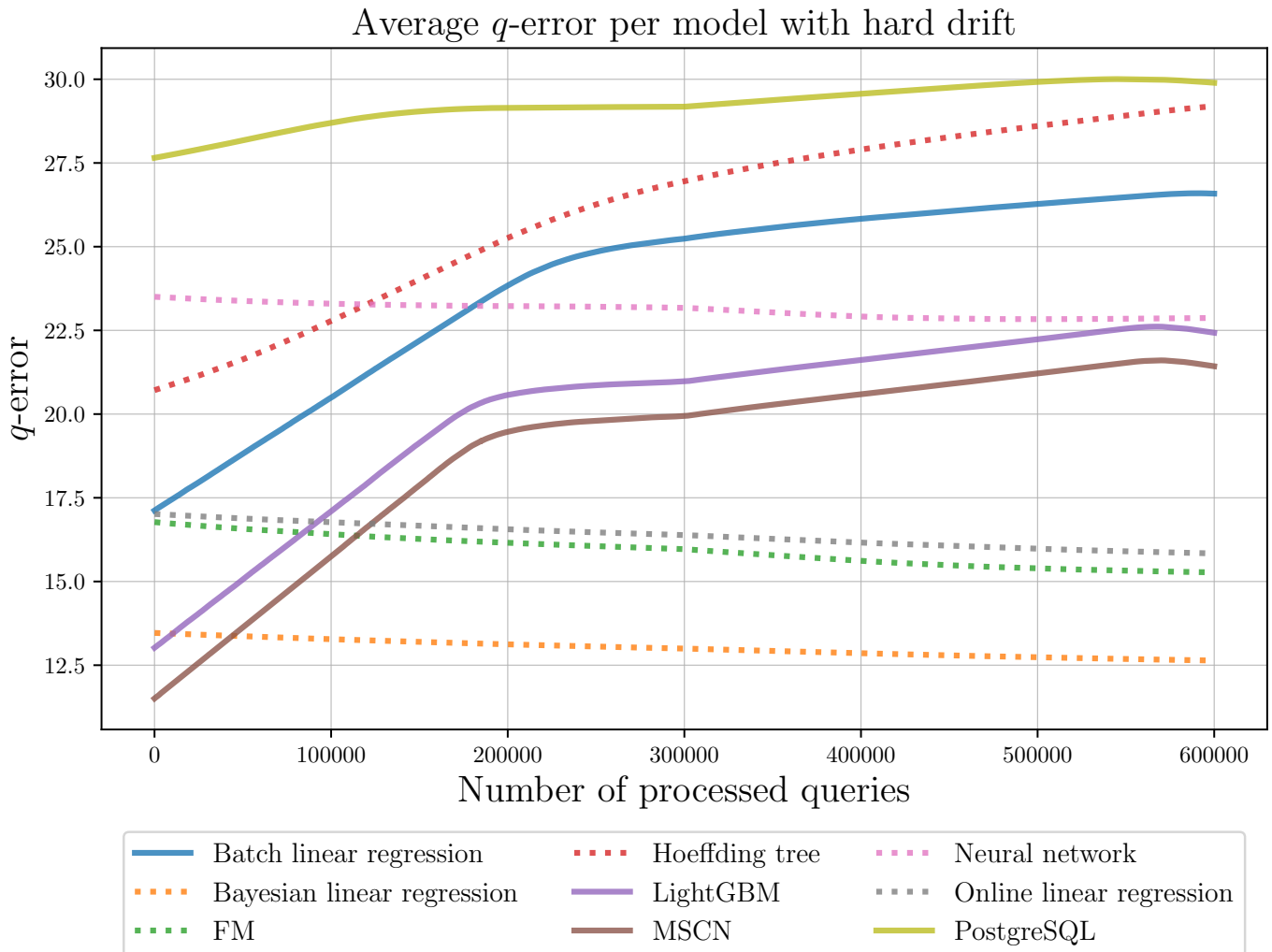


Figure: q-errors for each method with **soft** concept drift

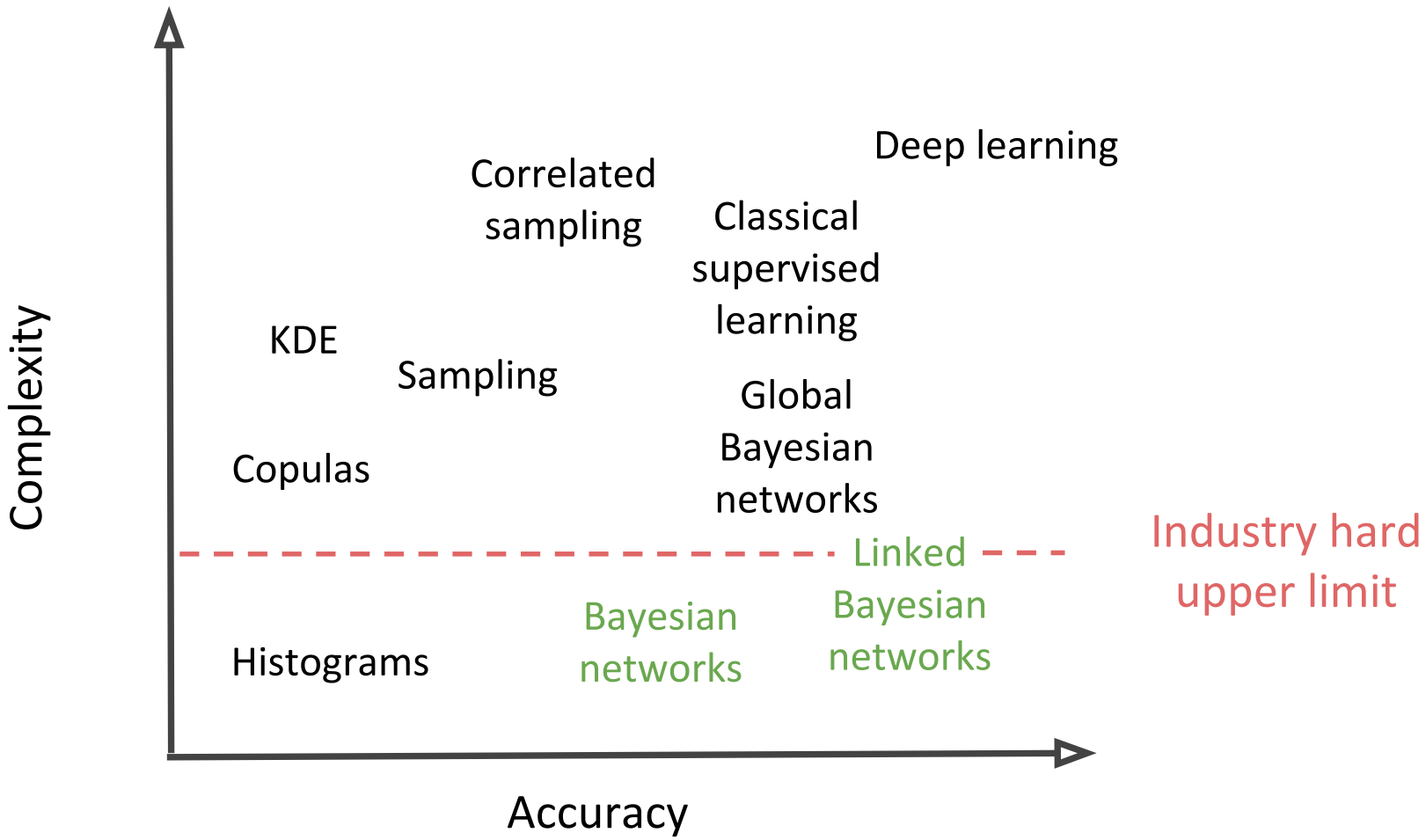


Conclusion

Conclusion

- Selectivity estimation is the backbone of query optimisation
- Computational performance is paramount, and thus dictates what we're allowed to consider
- Bayesian networks simplify the assumptions we presented above, without requiring too much computational resources
- Likewise, our online learning proposal focuses on simplicity

Offering a better compromise



Perspectives

1. Robust query optimisation
2. Extend to other applications:
   - Workload forecasting
   - Resource allocation
   - Approximate query answering
3. Implement in an actual query optimiser
4. Produce more benchmarks


Thank
you!

References I


Mert Akdere, Ugur Çetintemel, Matteo Riondato, Eli Upfal, and Stanley B Zdonik.
Learning-based query performance modeling and prediction.
In *2012 IEEE 28th International Conference on Data Engineering*, pages 390–401. IEEE, 2012.


Dariusz Rafal Augustyn.
Copula-based module for selectivity estimation of multidimensional range queries.
In *International Conference on Man–Machine Interactions*, pages 569–580. Springer, 2017.


Robert G Cowell, Philip Dawid, Steffen L Lauritzen, and David J Spiegelhalter.
Probabilistic networks and expert systems: Exact computational methods for Bayesian networks.
Springer Science & Business Media, 2006.

References II


C Chow and Cong Liu.
Approximating discrete probability distributions with dependence trees.
IEEE transactions on Information Theory, 14(3):462–467, 1968.


Chungmin Melvin Chen and Nick Roussopoulos.
Adaptive selectivity estimation using query feedback.
In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 161–172, 1994.


Pedro Domingos and Geoff Hulten.
Mining high-speed data streams.
In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

References III

Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri.
Selectivity estimation for range predicates using lightweight models.
Proceedings of the VLDB Endowment, 12(9):1044–1057, 2019.

Anshuman Dutt, Chi Wang, Vivek Narasayya, and Surajit Chaudhuri.
Efficiently approximating selectivity functions using low overhead regression models.
Proceedings of the VLDB Endowment, 13(11).

References IV


Dimitrios Gunopulos, George Kollios, J Tsotras, and Carlotta Domeniconi.
Selectivity estimators for multidimensional range queries over real attributes.
The VLDB Journal—The International Journal on Very Large Data Bases, 14(2):137–154, 2005.


Lise Getoor, Benjamin Taskar, and Daphne Koller.
Selectivity estimation using probabilistic models.
In *ACM SIGMOD Record*, volume 30, pages 461–472. ACM, 2001.

References V


Max Heimel, Martin Kiefer, and Volker Markl.
Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation.
In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1477–1492. ACM, 2015.


Frank K Hwang, Dana S Richards, and Pawel Winter.
The Steiner tree problem, volume 53.
Elsevier, 1992.


Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig.
Deepdb: learn from data, not from queries!
arXiv preprint arXiv:1909.00607, 2019.

References VI


Max Halford, Philippe Saint-Pierre, and Franck Morvan.
An approach based on bayesian networks for query selectivity estimation.
In *International Conference on Database Systems for Advanced Applications*, pages 3–19. Springer, 2019.


Max Halford, Philippe Saint-Pierre, and Franck Morvan.
Selectivity estimation with attribute value dependencies using linked bayesian networks.
arXiv preprint arXiv:2009.09883, 2020.


Oleg Ivanov and Sergey Bartunov.
Adaptive cardinality estimation, 2017.


Yannis E Ioannidis and Stavros Christodoulakis.
On the propagation of errors in the size of join results, volume 20.
ACM, 1991.

References VII


Yannis E Ioannidis and Stavros Christodoulakis.
Optimal histograms for limiting worst-case error propagation in the size of join results.
ACM Transactions on Database Systems (TODS), 18(4):709–748, 1993.


Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper.
Learned cardinalities: Estimating correlated joins with deep learning.
arXiv preprint arXiv:1809.00677, 2018.


Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu.
Lightgbm: A highly efficient gradient boosting decision tree.
In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

References VIII

Andreas Kipf, Dimitri Vorona, Jonas Müller, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, Thomas Neumann, and Alfons Kemper.
Estimating cardinalities with deep sketches.
In *Proceedings of the 2019 International Conference on Management of Data*, pages 1937–1940, 2019.

Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann.
How good are query optimizers, really?
Proceedings of the VLDB Endowment, 9(3):204–215, 2015.

References IX


Kukjin Lee, Arnd Christian König, Vivek Narasayya, Bolin Ding, Surajit Chaudhuri, Brent Ellwein, Alexey Eksarevskiy, Manbeen Kohli, Jacob Wyant, Praneeta Prakash, et al.
Operator and query progress estimation in microsoft sql server live query statistics.
In *Proceedings of the 2016 International Conference on Management of Data*, pages 1753–1764, 2016.


Richard J Lipton and Jeffrey F Naughton.
Query size estimation by adaptive sampling.
In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 40–46. ACM, 1990.

References X

Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann.
Query optimization through the looking glass, and what we found running the join order benchmark.
The VLDB Journal, pages 1–26, 2018.

Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, and Calisto Zuzarte.
Cardinality estimation using neural networks.
In *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, pages 53–59. IBM Corp., 2015.

Frank Olken.
Random sampling from databases.
PhD thesis, University of California, Berkeley, 1993.

References XI


Frank Olken and Doron Rotem.
Simple random sampling from relational databases.
1986.


Gregory Piatetsky-Shapiro and Charles Connell.
Accurate estimation of the number of tuples satisfying a condition.
ACM Sigmod Record, 14(2):256–276, 1984.


Steffen Rendle.
Factorization machines.
In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.


Neil Robertson and Paul D. Seymour.
Graph minors. ii. algorithmic aspects of tree-width.
Journal of algorithms, 7(3):309–322, 1986.

References XII


P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price.
Access path selection in a relational database management system.
In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, pages 23–34, 1979.


Michael Stillger, Guy M Lohman, Volker Markl, and Mokhtar Kandil.
Leo-db2's learning optimizer.
In *VLDB*, volume 1, pages 19–28, 2001.


Hien To, Kuorong Chiang, and Cyrus Shahabi.
Entropy-based histograms for selectivity estimation.
In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1939–1948. ACM, 2013.

References XIII


Kostas Tzoumas, Amol Deshpande, and Christian S Jensen.
Lightweight graphical models for selectivity estimation without independence assumptions.
Proceedings of the VLDB Endowment, 4(11):852–863, 2011.


David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P Chakkappen.
Join size estimation subject to filter conditions.
Proceedings of the VLDB Endowment, 8(12):1530–1541, 2015.


Wentao Wu, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Hakan Hacigümüs, and Jeffrey F Naughton.
Predicting query execution time: Are optimizer cost models really unusable?
In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1081–1092. IEEE, 2013.

References XIV

Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner.
Cardinality estimation with local deep learning models.
In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, page 5. ACM, 2019.

Chenggang Wu, Alekh Jindal, Saeed Amizadeh, Hiren Patel, Wangchao Le, Shi Qiao, and Sriram Rao.
Towards a learning optimizer for shared clouds.
Proceedings of the VLDB Endowment, 12(3):210–222, 2018.

References XV


Yaoshu Wang, Chuan Xiao, Jianbin Qin, Xin Cao, Yifang Sun, Wei Wang, and Makoto Onizuka.
Monotonic cardinality estimation of similarity selection: A deep learning approach.
In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1197–1212, 2020.


Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi.
Random sampling over joins revisited.
2018.