FAST ALGORITHMS FOR MINING ASSOCIATION RULES

Rakesh Agrawal

Ramakrishnan Srikant

IBM Almaden Research Center

Presenter: George Du

Discussion Leader: Marie Salomon

BASKET DATA

- Definition: massive amounts of sales data collected by the retail organizations (e.g. Transaction data including sales date + items in the transaction).
- Basket data can be used to implement customized marketing programs and strategies.
- Dataset is often large, hence efficient algorithm is required to analyze the data.

ASSOCIATION RULES

- One type of the rules that can be mined from basket data & the problem being focused in this paper.
- Example: 98% of customers getting tires and auto accessories also get automotive service done.
- This insight is important because the retail organization can utilize this information to increase sales.

ASSOCIATION RULES

- X=>Y
  - I: Set of items $\{i_1, i_2, \ldots, i_m\}$
  - D: Set of transactions
  - T: Transaction with a unique TID, $T \subseteq I$
  - $X \subset I$, $Y \subset I$, $X \cap Y = \emptyset$

ASSOCIATION RULES - CONFIDENCE

- X=>Y
  - Confidence c: c% of transactions in D that contain X also contain Y
  - c = countOf(X+Y)/countOf(X)

ASSOCIATION RULES - SUPPORT

- X=>Y
  - Support s: s% of transactions in D contain X + Y
  - s = countOf(X+Y)/size(D)

MINING ASSOCIATION RULES

- Goal: Given a set of transactions D, generate all association rules that have support and confidence greater than the user-specified minimum support (minsup) and minimum confidence (minconf)
- Algorithms used: Apriori and AprioriTid

SUBPROBLEMS

1. Find all sets of items (itemsets) that have transaction support above minsup (referred to as large itemsets).
   - The focus of this paper
2. Use the large itemsets to generate the desired rules.
   - Not being discussed in this paper

DISCUSSION – GROUPS OF 2

- In the comparison of algorithms presented in the paper, various factors such as performance efficiency, scalability, and applicability to real-world data sets are considered.
    - What criteria do you believe are most critical when evaluating algorithms?
    - What should we consider when evaluating an algorithm to provide a fair and comprehensive comparison with other work?

APRIORI

- Basic intuition: Any subset of a large itemset must be large
- Therefore, the candidate itemsets having k items can be generated by joining large itemsets having k - 1 items, and deleting those that contain any subset that is not large.
- Result: much smaller number of candidate itemsets, more efficient algorithm.

APRIORI

- First pass: count the support of individual items, find the large items
- Each subsequent pass: start with a seed set of itemsets found to be large in the previous pass, generate new candidate itemsets, count the actual support, find large itemsets and use as seed for next pass
- Continue: until no new large itemsets are found

APRIORI – PSEUDO CODE

1) $L_1 = \{\text{large 1-itemsets}\}$;
2) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
3) $C_k = \text{apriori-gen}(L_{k-1})$; // New candidates
4) **forall** transactions $t \in \mathcal{D}$ **do begin**
5) $C_t = \text{subset}(C_k, t)$; // Candidates contained in t
6) **forall** candidates $c \in C_t$ **do**
7) $c.\text{count}$++;
8) **end**
9) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
10) **end**
11) Answer $= \bigcup_k L_k$;

APRIORI – CANDIDATE GENERATION

- Join: L_{k-1} with L_{k-1}

```
insert into C_k
select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
from L_{k-1} p, L_{k-1} q
where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},
      p.item_{k-1} < q.item_{k-1};
```

- Prune: delete itemsets with (k-1)-subset that is not in L_{k-1}

```
forall itemsets c ∈ C_k do
  forall (k-1)-subsets s of c do
    if (s ∉ L_{k-1}) then
      delete c from C_k;
```

APRIORI - EXAMPLE

- First pass: find large items, minsup = 0.5
- {1}: s = 2/4
- {2}: s = 3/4
- {3}: s = 3/4
- ~~{4}: s = 1/4~~
- {5}: s = 3/4

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORI - EXAMPLE

- Second pass: find large 2-itemsets
- Candidates:
  - {1, 2}
  - {1, 3}
  - ~~{1, 4}~~
  - {1, 5}
  - {2, 3}
  - ~~{2, 4}~~
  - {2, 5}
  - ~~{3, 4}~~
  - {3, 5}
  - ~~{4, 5}~~

Itemsets containing item 4 are pruned since {4} is not large!

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORI - EXAMPLE

- Second pass: find large 2-itemsets
- Candidates:
  - ~~{1, 2}: s = 1/4~~
  - {1, 3}: s = 2/4
  - ~~{1, 5}: s = 1/4~~
  - {2, 3}: s = 2/4
  - {2, 5}: s = 3/4
  - {3, 5}: s = 2/4

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORI - EXAMPLE

- Third pass: find large 3-itemsets
- Candidates:
  - ~~{1, 2, 3}~~ Pruned because {1, 2} is not large
  - ~~{1, 2, 5}~~ Pruned because {1, 2} is not large
  - ~~{1, 3, 5}~~ Pruned because {1, 5} is not large
  - {2, 3, 5}

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORI - EXAMPLE

- Third pass: find large 3-itemsets
- Candidates:
  - {2, 3, 5}: s = 2/4

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORI - EXAMPLE

- Resulting large itemsets:
  - {1}
  - {2}
  - {3}
  - {5}
  - {1, 3}
  - {2, 3}
  - {2, 5}
  - {3, 5}
  - {2, 3, 5}

No more new itemsets can be identified, end execution!

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

APRIORITID

- Same way to generate set of candidates using the apriori-gen function.
- Difference is that it does not use database D for computing support.
- Instead, it uses $\overline{C}_k$, in the form of <TID, $\{X_k\}$>, where each X_k is a potentially large k-itemset present in the transaction with identifier TID.
- For k = 1, $\overline{C}_k$ corresponds to the database D.

APRIORITID – PSEUDO CODE

1) $L_1 = \{\text{large 1-itemsets}\}$;
2) $\overline{C}_1 = \text{database } \mathcal{D}$;
3) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
4) $C_k = \text{apriori-gen}(L_{k-1})$; // New candidates
5) $\overline{C}_k = \emptyset$;
6) **forall** entries $t \in \overline{C}_{k-1}$ **do begin**
7) // determine candidate itemsets in C_k contained
// in the transaction with identifier t.TID
$C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{set-of-itemsets} \wedge$
$(c - c[k-1]) \in t.\text{set-of-itemsets}\}$;
8) **forall** candidates $c \in C_t$ **do**
9) c.count++;
10) **if** $(C_t \neq \emptyset)$ **then** $\overline{C}_k$ += $< t.\text{TID}, C_t >$;
11) **end**
12) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
13) **end**
14) Answer = $\bigcup_k L_k$;

Notice how D is replaced with $\overline{C}_k$ here.

APRIORITID - EXAMPLE

L_1

Itemset	Support
{ 1 }	2
{ 2 }	3
{ 3 }	3
{ 5 }	3

$\overline{C}_1$

TID	Set-of-itemsets
100	{ {1}, {3}, {4} }
200	{ {2}, {3}, {5} }
300	{ {1}, {2}, {3} }, {5} }
400	{ {2}, {5} }

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Minsup = 2

APRIORITID - EXAMPLE

C_2

Itemset	Support
{ 1 2 }	1
{ 1 3 }	2
{ 1 5 }	1
{ 2 3 }	2
{ 2 5 }	3
{ 3 5 }	2

L_2

Itemset	Support
{ 1 3 }	2
{ 2 3 }	2
{ 2 5 }	3
{ 3 5 }	2

$\overline{C}_1$

TID	Set-of-itemsets
100	{ {1}, {3}, {4} }
200	{ {2}, {3}, {5} }
300	{ {1}, {2}, {3} }, {5} }
400	{ {2}, {5} }

Minsup = 2

APRIORITID - EXAMPLE

C_3

Itemset	Support
{ 2 3 5 }	2

L_3

Itemset	Support
{ 2 3 5 }	2

$\overline{C}_3$

TID	Set-of-itemsets
200	{ { 2 3 5 } }
300	{ { 2 3 5 } }

$\overline{C}_2$

TID	Set-of-itemsets
100	{ {1 3} }
200	{ {2 3}, {2 5}, {3 5} }
300	{ {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5} }
400	{ {2 5} }

Minsup = 2

PERFORMANCE – SAMPLE DATA

Name	$\|T\|$	$\|I\|$	$\|D\|$	Size in Megabytes
T5.I2.D100K	5	2	100K	2.4
T10.I2.D100K	10	2	100K	4.4
T10.I4.D100K	10	4	100K	
T20.I2.D100K	20	2	100K	8.4
T20.I4.D100K	20	4	100K	
T20.I6.D100K	20	6	100K	

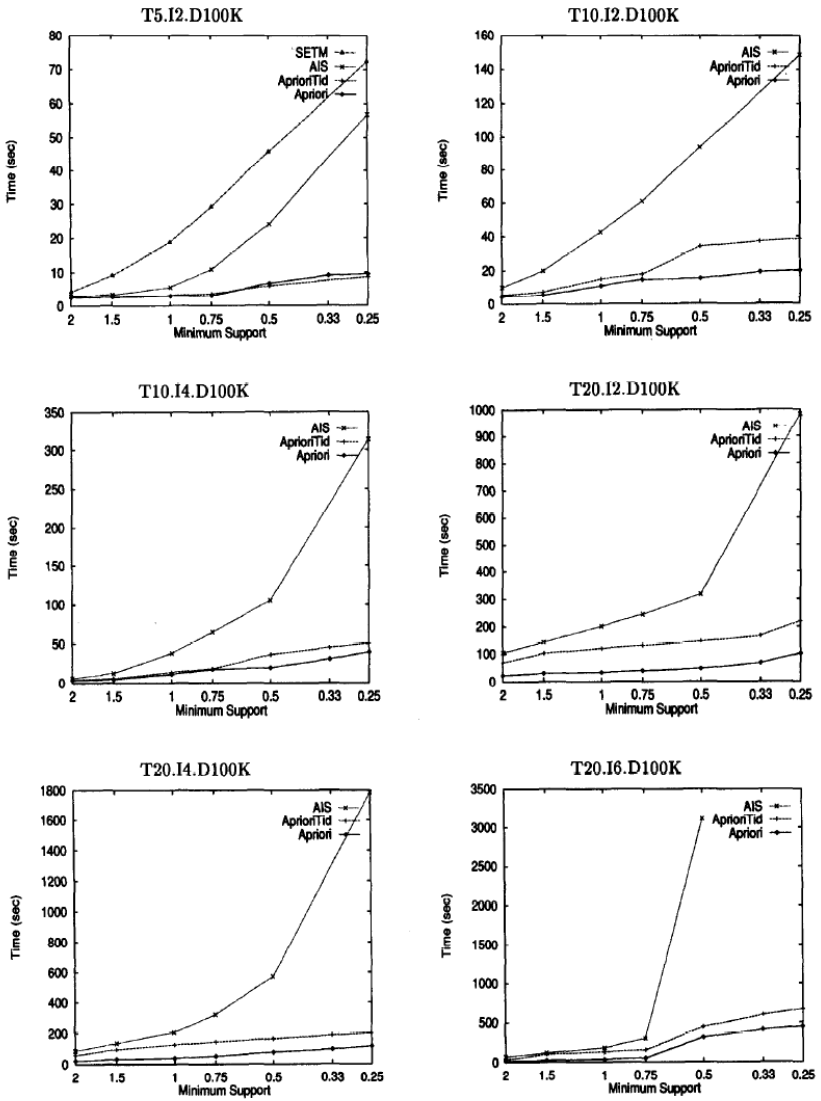
- |D|: Number of transactions
- |T|: Average size of the transactions
- |I|: Average size of the maximal potentially large itemsets
- |L|: Number of maximal potentially large itemsets
- N: Number of items

PERFORMANCE - COMPARISON

Table 4: Execution times in seconds for SETM

Algorithm	Minimum Support				
	2.0%	1.5%	1.0%	0.75%	0.5%
Dataset T10.I2.D100K					
SETM	74	161	838	1262	1878
Apriori	4.4	5.3	11.0	14.5	15.3
Dataset T10.I4.D100K					
SETM	41	91	659	929	1639
Apriori	3.8	4.8	11.2	17.4	19.3

- AprioriTid's performance degraded to about twice as slow for large problems.
- Apriori and AprioriTid outperforms SETM and AIS, and the difference gets bigger when minimum support decreases
- Apriori beats AIS for all problem sizes, by factors ranging from 2 for high minimum support to more than an order of magnitude for low levels of support.



APRIORIHYBRID

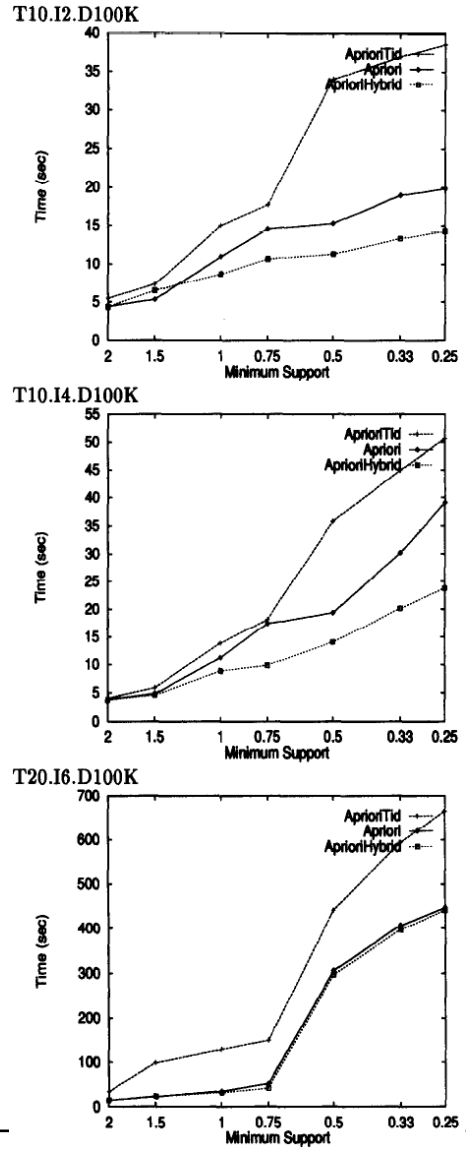
- Observation:

  In the earlier passes, Apriori does better than AprioriTid. However, AprioriTid beats Apriori in later passes.

- Reason:
  - Both algorithms use the same candidate generation procedure.
  - In the later passes, # of candidate itemsets reduces, but Apriori still examines every transaction in the database.
  - AprioriTid scans $\overline{C}_k$ for support counts, which becomes smaller than the size of the database. When it fits in memory, can skip writing to disk.

APRIORIHYBRID

- Use Apriori in initial passes
- Switch to AprioriTid when $\overline{C}_k$ is expected to fit in memory
- AprioriHybrid performs better than Apriori in almost all cases
- Advantage depends on how the size of $\overline{C}_k$ decline, if there's a gradual decline, AprioriTid kicks-in sooner, resulting in significant improvement in execution time



DISCUSSION – GROUPS OF 4

- This paper has had a big influence on the database community spinning off more similar algorithms in the database world than any other data mining algorithm. Interestingly the paper was motivated by a very particular business task. What factors contribute to this high level of academic and practical interest?
    - The research topic and problem?
    - Is the algorithm's design, efficiency, or novelty a key driver of its popularity?
    - The methodology used for developing and assessing the solution?
    - Or other factors?

SUMMARY

- New algorithms: Apriori, AprioriTid, and a combination of two (AprioriHybrid)
- Performance measurement: All better than AIS and SETM
- Execution time difference among them depends on characteristic of dataset

BIAS IN OLAP QUERIES: DETECTION, EXPLANATION, AND REMOVAL

Babak Salimi
Johannes Gehrke
Dan Suciu

University of Washington
Microsoft

Presenter: George Du

Discussion Leader: Marie Salomon

ON LINE ANALYTICAL PROCESSING - OLAP

- Essential element of decision-support systems
- Complex calculations, analyses, and sophisticated data modeling, aiming to provide the insights and understanding needed for improved decision making

BIASED QUERY

- Unfortunately, bad queries can result in perplexing insights, leading to poor business decisions!

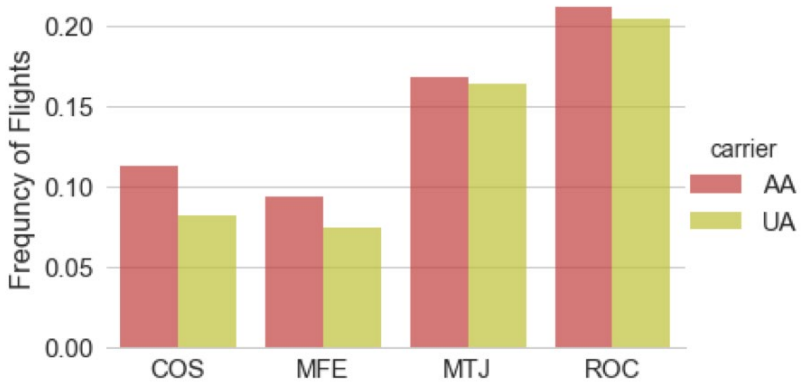
BIASED QUERY – EXAMPLE

- A company wants to choose between the business travel programs offered by two carriers, American Airlines (AA) and United Airlines (UA).
- Four airports: Rochester (ROC), Montrose (MTJ), McAllen Miller (MFE) and Colorado Springs (COS).
- The company wants to choose the carrier with the lowest rate of delay at these airports.

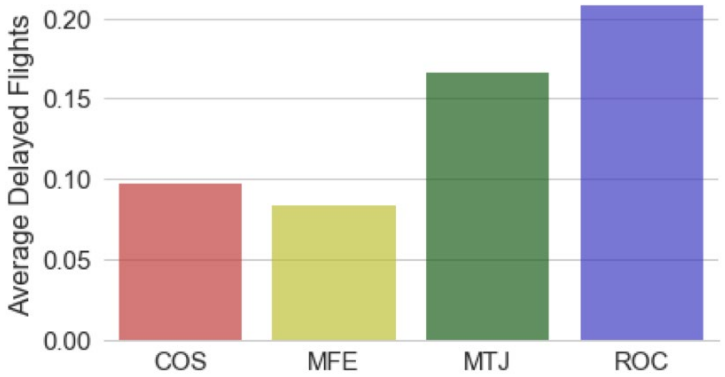
BIASED QUERY – EXAMPLE

- Data analyst uses FlightData, the historical flight data
- Runs query to make decision

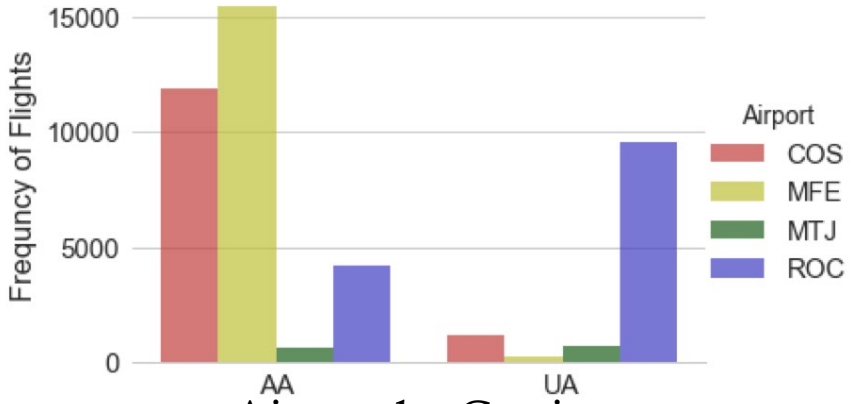
Database Schema:
FlightData(Year, Quarter, Dayofweek, Airport, Dest, DepartureTime, Carrier, Delayed, …)



Carriers Delay by Airport



Delay by Airport

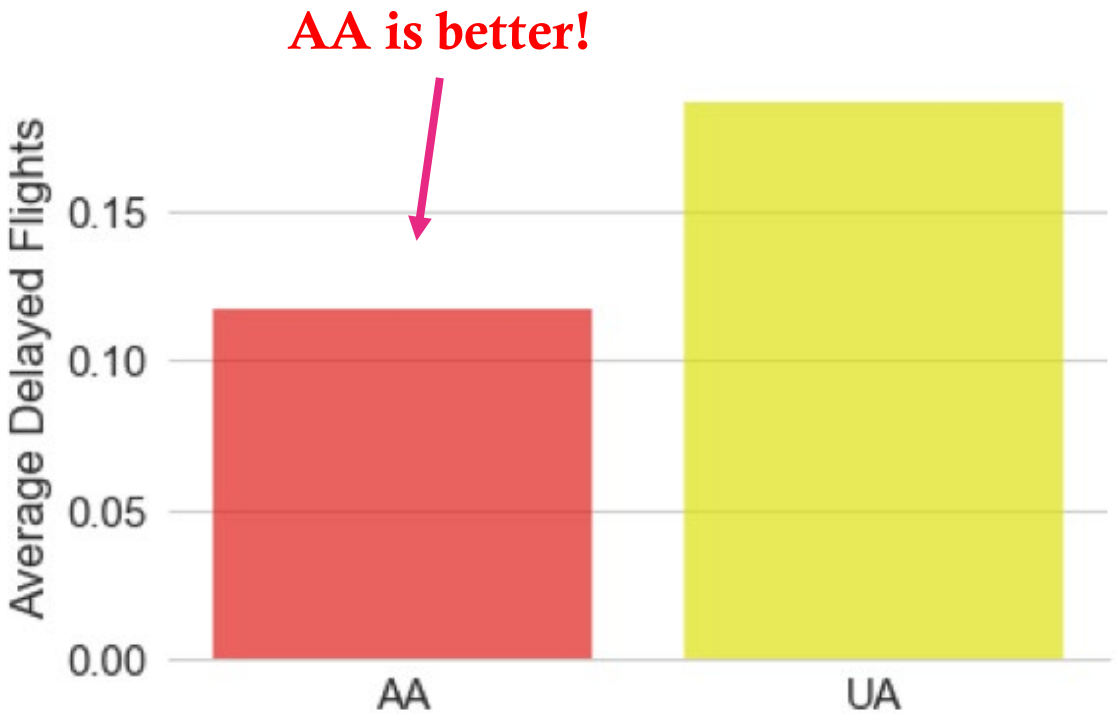


Airport by Carrier

BIASED QUERY – EXAMPLE

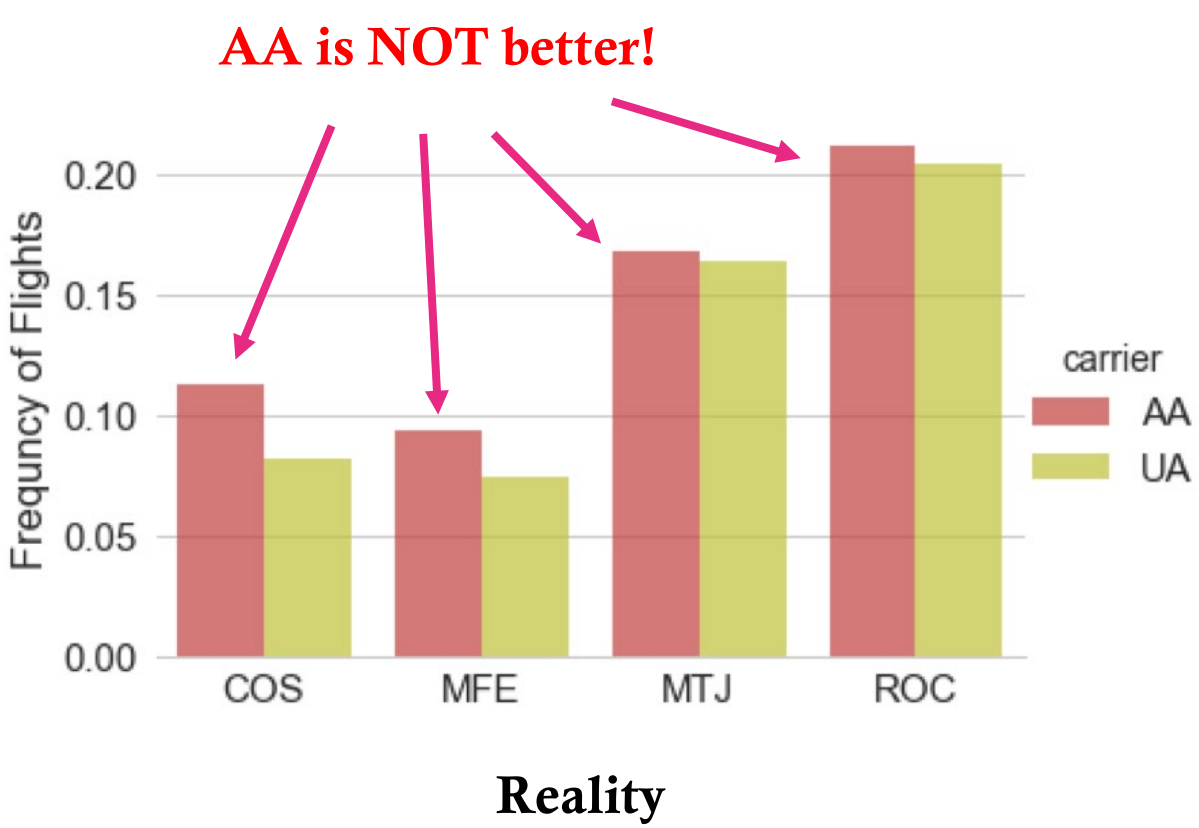
- OLAP Query:

SELECT avg(Delayed)
FROM FlightData
GROUP BY Carrier
WHERE Carrier IN ('AA','UA')
AND Airport IN
('COS','MFE','MTJ','ROC')



Query Answer

BIASED QUERY – EXAMPLE



Reality

Query Answer

DISCUSSION – GROUPS OF 4

- Have you heard of any other examples in computer science or even your own research where biased was introduced?
  - What are the causes of the bias?
  - How was the bias detected?
  - What are our responsibilities as computer scientists when it comes to releasing a system or algorithm that is biased?

BIASED QUERY – WHAT WENT WRONG?

- AA has many more flights from airports that have relatively few delays, like COS and MFE, while UA has more flights from ROC, which has relatively many delays.
- AA seems to have overall lower delay only because it has many flights from airports that in general have few delays.
- Incorrect interpretation of the query: while the analyst's goal is to compare the causal effect of the carriers on delay, the OLAP query measures only their association.

BIASED QUERY – SIMPSON'S PARADOX

- A phenomenon in probability and statistics in which a trend appears in several groups of data but disappears or reverses when the groups are combined

PROBLEM IN OLAP SYSTEMS

- No causal analysis tools exist for OLAP systems
- Most analysts use group-by queries, prone to biased business decisions

HYPDB

- Detect

  A new formal definition of a biased query enables the system to detect bias in OLAP queries by performing a set of independence tests on the data

- Explain

  Novel technique to find explanations for the bias and to rank these explanations, assisting the analyst in understanding what goes on.

- Resolve

  Query rewriting technique to eliminate the bias from queries

CORE OF CAUSAL ANALYSIS - COVARIATE

- An attribute that is correlated ("covaries") with the outcome and unaffected by the treatment
- Example: A Group By T query is biased if there exists a set of covariates **Z** whose distribution differs in different groups of T. Like the Airport in the previous example.
- To draw causal conclusions from a biased query, one needs to control for covariates and thereby to eliminate other possible explanations for a causal connection indicated by the query

BACKGROUND

Listing 1: An OLAP query Q.

```
SELECT T,X,avg(Y1), ... ,avg(Ye)
FROM D
WHERE C
GROUP BY T,X
```

- Sets of attributes: **X**
- Treatment variable: T, Dom(T) = {t0, t1}
- Outcome variable: Y, Dom(Y) = {0, 1}
- Context of the query: $\Gamma_i \stackrel{\text{def}}{=} \mathbf{C} \wedge (\mathbf{X} = \mathbf{x}_i)$
- Set of covariates: **Z**, subset of all attributes

ASSUMPTIONS

- For all $z \in Dom(\mathbf{Z})$
  1. $(Y(t0), Y(t1) \perp\!\!\!\perp T \mid \mathbf{Z} = z)$

     Unconfoundedness: Given $\mathbf{Z} = z$, the outcomes of $Y(t0)$ and $Y(t1)$ are conditionally independent of treatment T.
  2. $0 < Pr(T = t1 \mid \mathbf{Z} = z) < 1$

     Overlap: Given $\mathbf{Z} = z$, the propability distribution of $T = t1$ is greater than 0 and smaller than 1.

DETECTING BIAS

- Check $(T \perp\!\!\!\perp \mathbf{V} | \Gamma_i)$

  A query Q is balanced w.r.t. a set of variables **V** in a context Γi if the marginal distributions Pr(**V** | T = t0, Γi) and Pr(**V** | T = t1, Γi) are the same, meaning the distribution of **V** is not dependent on the treatment value, ensuring that T is not confounded by **V**.

- If query is balanced, the groups are comparable in every relevant respect, i.e., the distribution of potential covariates such as age, proportion of male/female, qualifications, motivation, experience, abilities, etc., are similar in all groups.

EXPLAINING BIAS

- Coarse-grained

  Ranking the variables **V**, in terms of their responsibilities for the bias

- Fine-grained

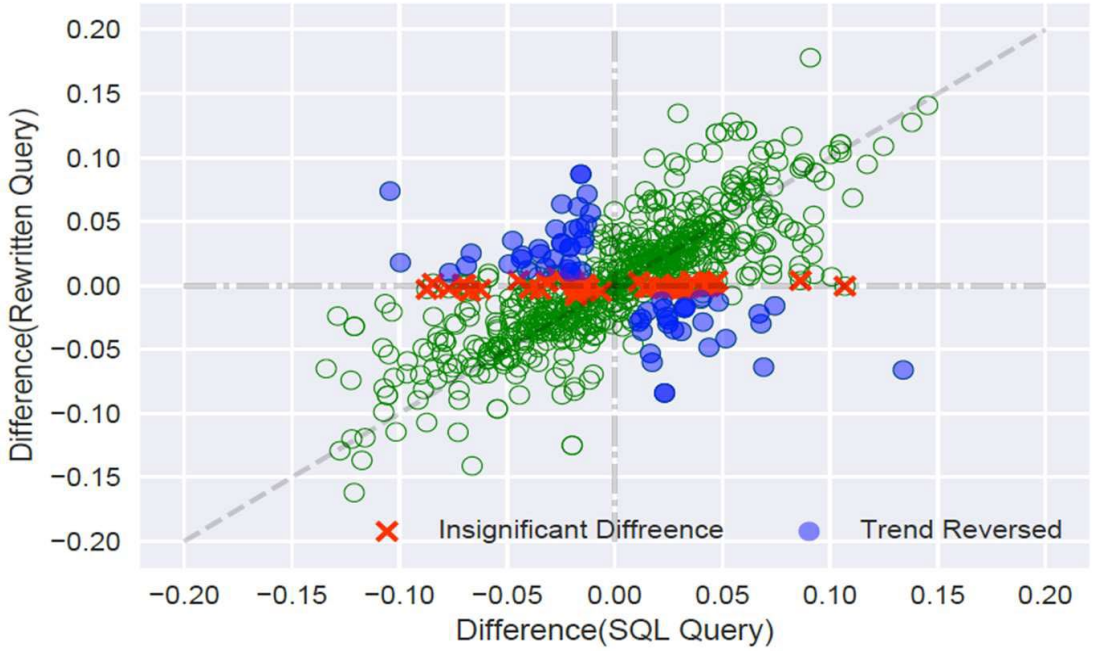
  Providing a triple (t, y, z), where t ∈ Dom(T), y ∈ Dom(Y), z ∈ Dom(Z), that highly contributes to both I (T ;Z) and I (Y;Z) (mutual information). These triples explain the confounding (or mediating) relationships between the ground levels.

RESOLVING BIAS

- Partitions the data into blocks that are homogeneous on **Z**.
- Computes the average of each Y ∈ **Y** Group by T, **X**, in each block.
- Aggregates the block's averages by taking their weighted average, where the weights are probabilities of the blocks.
- To enforce Overlap, discard all blocks that do not have at least one tuple with T=t1 and one tuple with T=t2.
- This removes the bias introduced by the covariates **Z** by conditioning on it.

EXPERIMENTS – AVOIDING FALSE DISCOVERIES

- To what extent HypDB does prevent the chance of false discoveries?
- Generated 1000 random queries.
- Used HypDB to rewrite queries w.r.t. the potential covariates.
- 20% of the cases, query rewriting reversed the trend.



a) The effect of query rewriting on FlightData.

EXPERIMENTS – END-TO-END RESULTS

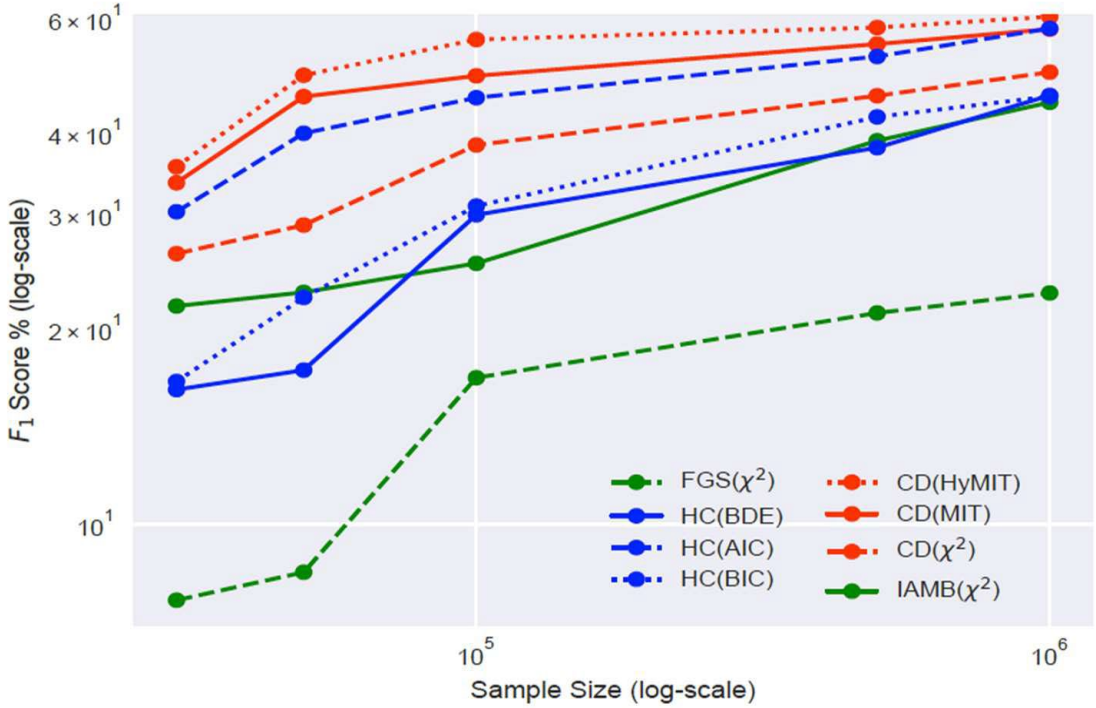
- Performed end-to-end tests on five different datasets, reported the runtime in seconds for each individual dataset.

Dataset	Columns [#]	Rows[#]	Det.	Exp.	Res.
AdultData [22]	15	48842	65	<1	<1
StaplesData [49]	6	988871	5	<1	<1
BerkeleyData [3]	3	4428	2	<1	<1
CancerData [15]	12	2000	<1	<1	<1
FlightData [42]	101	43853	20	<1	<1

Table 1: Runtime in seconds for experiments in Sec. 7.3.

EXPERIMENTS – QUALITY COMPARISON

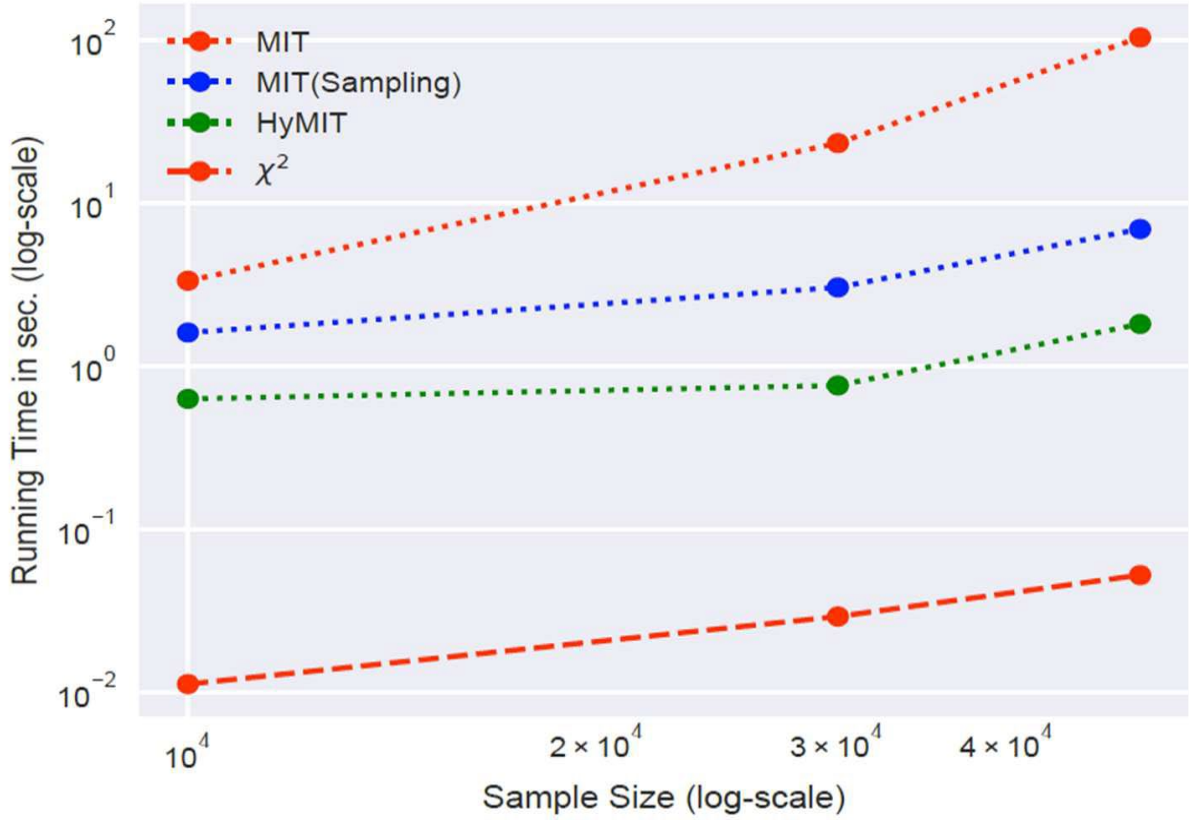
- Comparison with other algorithms shows that the algorithm significantly outperforms most other algorithms.



b) Quality comparison.

EXPERIMENTS – EFFICACY OF OPTIMIZATION TECHNIQUES

- evaluate the quality of the optimizations proposed for nonparametric independence tests
- MIT with sampling and HyMIT are much faster that MIT



b) Efficacy of the optimizations proposed for independence tests.

DISCUSSION – GROUPS OF 2

- How can people be more sensitized to detect bias?
- What needs to change (e.g. when teaching students) to educate people about the importance of ethical computing? Should discussing ethics be mandatory? Should we teach it in specialized courses? In all courses? Both?
- What could be done to reduce the bias in computer science?

SUMMARY

- Proposed HypDB, a system to detect, explain, and resolve bias in decision-support OLAP queries.
- Showed that biased queries can be perplexing and lead to statistical anomalies, such as Simpson's paradox.
- Presented details of HypDB's design, and experimentation results.