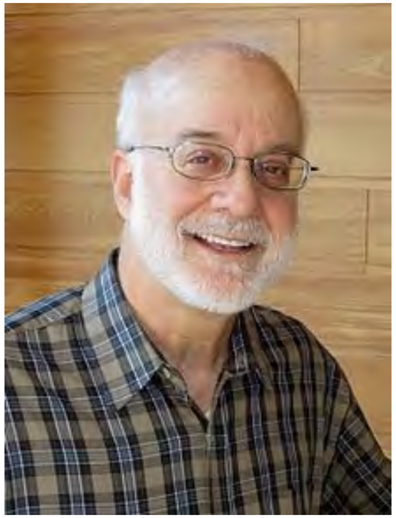
Optimal Aggregation Algorithms for Middleware

Ron Fagin Amnon Lotem Moni Naor

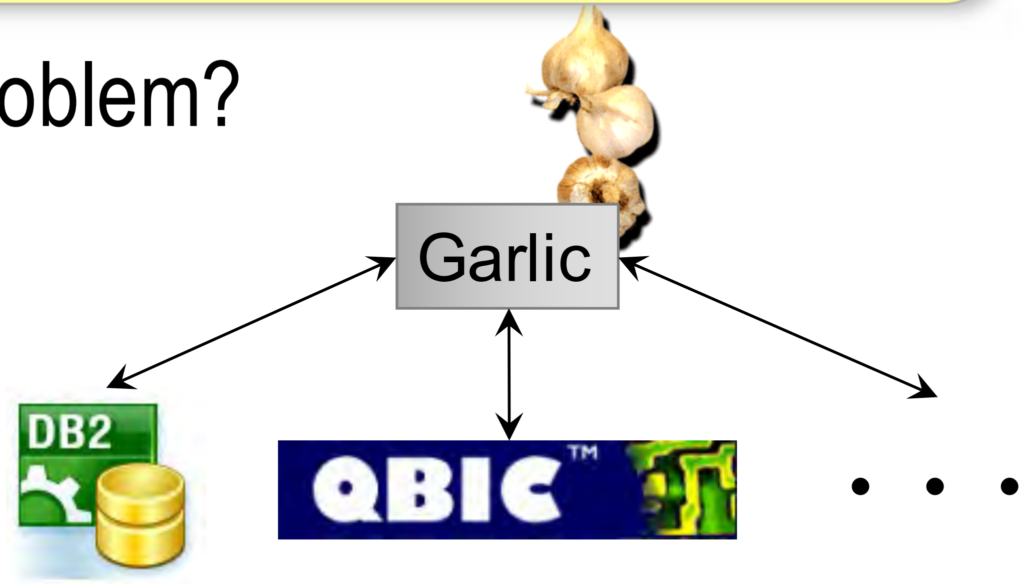


Gems of PODS talk, 2016

Laura Haas

Mr. Database Theoretician, we've got a problem with Garlic, our multimedia database system!

- What was the problem?



Example databases:

- The answers to queries in DB/2 are sets
- The answers to queries in QBIC are sorted lists
- How do you combine the results?

Example

- Searching a CD database for Artist = “Beatles” yields a set, via, say DB/2

Musicbrainz has 12 million recordings in its DB

Example

- AlbumColor = “Red” yields a sorted list, via, say QBIC

.697

.683

.670

.659

.629

Redness

Example

- How do we make sense of

  $(\text{Artist} = \text{'Beatles'}) \wedge (\text{AlbumColor} = \text{'Red'})$?

  – Here it is probably a list of albums by the Beatles, sorted by how red they are

- What about

  $(\text{Artist} = \text{'Beatles'}) \vee (\text{AlbumColor} = \text{'Red'})$?

- And what about

  $(\text{Color} = \text{'Red'}) \wedge (\text{Shape} = \text{'Round'})$?

What Was My Solution?

- These weren't just sorted lists: they were ***scored*** *lists*
- Can view sets as scored lists (scores 0 or 1)
- This reminded me of fuzzy logic
- In fuzzy logic, conjunction ($\wedge$) is min, and disjunction ($\vee$) is max

Use fuzzy logic

I like your solution. But we also need an efficient algorithm that can find the top k results while minimizing database accesses

⋮

Laura Haas

I have an algorithm that finds the top k with only √n database accesses

Ron Fagin

Good, that beats linear! But we database people are spoiled, and are used to only log n accesses. Be smarter and get me a log n algorithm

⋮

I proved that you can't do better than √n

Time for the Accesses

- Say n = 12,000,000 CDs
- Assume 1000 accesses per second
- n accesses (naïve algorithm) would take 3 hours
- $\sqrt{n}$ accesses would take 3 seconds

Generalizing the Algorithm

- The algorithm works for arbitrary monotone scoring functions
  - increasing the scores of arguments cannot decrease the overall score

The Problem

- There are *m* attributes, or fields
- Each object in a database has a score x_i for attribute *i*
- The objects are given in *m* sorted lists, one list per attribute
- Goal: Find the top *k* objects according to a monotone scoring function, while minimizing access to the lists

- Can think of the attributes as voters, and the objects as candidates, where each voter assigns a score to each candidate

Multimedia Example

REDNESS
177: 0.993
139: 0.991
702: 0.982
. . .
235: 0.325
. . .

ROUNDNESS
235: 0.999
666: 0.996
820: 0.992
. . .
177: 0.406
. . .

Scoring Functions

- Let f be the scoring function
- Popular choices for f:
  - min (used in fuzzy logic)
  - average
- Let $x_1, \ldots, x_m$ be the scores of object R under the m attributes
- Then $f(x_1, \ldots, x_m)$ is the overall score of object R
- A scoring function f is *monotone* if whenever

 $x_i \le y_i$ for every i, then $f(x_1, \ldots, x_m) \le f(y_1, \ldots, y_m)$

Modes of Access

- Sorted (or sequential) access
  – Can obtain the next object and its score for attribute *i*
- Random access
  – Can obtain the score of object *R* for attribute *i*
- Wish to minimize total number of accesses

Algorithms

- Want an algorithm for finding the top *k* objects
- Naïve algorithm retrieves every score of every object
  - Too expensive

Fagin's Algorithm - FA

Combining fuzzy information from multiple systems, PODS'96, JCSS'99

- For all lists $L_1, L_2, \ldots, L_m$ **get next** object in sorted order.
- **Stop** when there is set of k objects that appeared in **all** lists.
- For ***every*** object R encountered
  - retrieve **all** fields $x_1, x_2, \ldots, x_m$.
  - Compute $f(x_1,x_2,\ldots x_m)$
- Return top k objects

Correctness of the Halting Rule

Assume (by way of contradiction):

R unseen; S in top k ; $\boxed{f(R)>f(S)}$

Let $T_1, \ldots, T_k$ be the objects that appeared in every list.

Since S is in the top k, there is p s.t. $\boxed{f(S) \geq f(T_p).}$

So $f(R) > f(T_p)$.

Hence for some attribute j the score of R *on* attribute j is bigger than the score of T_p on attribute j.

Since T_p appeared in L_j under sorted access, so did R, which is a contradiction.

Performance of FA

Performance : assuming that the fields are **independent** $O(n^{(m-1)/m})$.

Under **independence** assumption, FA is optimal with high probability in the worst case for all "strict" scoring functions ("strict" means that the value is 1 iff all arguments are 1)

Influence

Algorithm implemented in Garlic

Influenced other IBM products, including

- Watson Bundled Search system
- InfoSphere Federation Server
- WebSphere Commerce

Paper introducing my algorithm has over 800 citations (Google Scholar)

Enter Amnon Lotem

Mike Franklin taught an advanced course in databases at the University of Maryland

- Autumn 1997

- Amnon Lotem was a student
- Mike suggested Amnon to read Fagin’s paper
- Amnon found an algorithm that was “better” than the “optimal” Fagin’s Algorithm
  - Convinced Mike, via simulations

Enter Moni Naor

1999-2001:

Sabbatical from Weizmann Institute

At

- Stanford

- IBM Almaden

Almaden Research Center

Stanford Advanced Image Databases

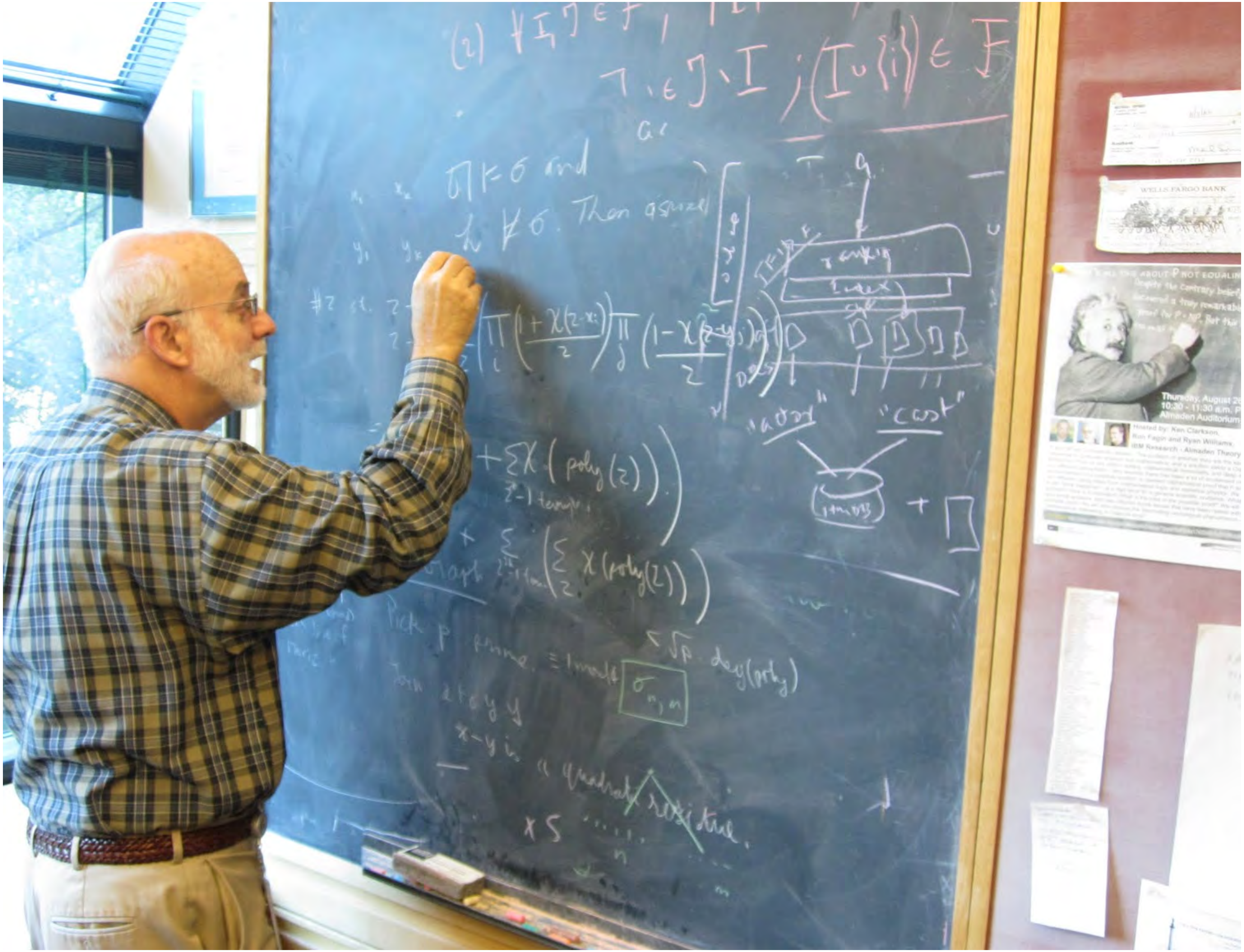
TIME:	Fridays, 3:15pm until 4:30pm. Please arrive 5 min. early to sign in!
LOCATION & DIRECTIONS:	201 T-Seq, right across the street from the Gates Information Sciences building
INFORMATION:	michel@CS.Stanford.EDU

Seminar Schedule

This quarter the talks will focus on ***Ontologies, E-Commerce, XML, and Metadata.***

8 October 1999	Fuzzy Queries in Multimedia Database Systems (slides in postscript)	*NAME*	Ron Fagin
		AFFILIATION	IBM Almaden Research Center

Source: http://i.stanford.edu/infoseminar/archive/FallY99/


WELLS FARGO BANK
Almaden Auditorium
Hosted by: Ken Clarkson,
Ron Fagin and Ryan Williams,
IBM Research - Almaden Theory

Threshold Algorithm

- Do sorted access in parallel to each of the *m* scored lists.
- As each object *R* is seen under sorted access:
  - Do random access to retrieve all of its scores $x_1, \ldots, x_m$
  - Compute its overall score $f(x_1, \ldots, x_m)$
  - If this is one of the top *k* answers so far, remember it
- For each list *i*, let t_i be the score of the last object seen under sorted access
- Define the threshold value *T* to be $f(t_1, \ldots, t_m)$. When *k* objects have been seen whose overall score is at least *T*, stop
- Return the top *k* answers

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999

Scoring function is min

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999
	. . .
	177: 0.406
	. . .

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999
. . .	. . .
235: 0.325	177: 0.406
. . .	. . .

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999
. . .	. . .
235: 0.325	177: 0.406
. . .	. . .

Overall score for 177: min(0.993, 0.406) = .406

Overall score for 235: min(0.325, 0.999) = .325

Threshold Algorithm: Example (using min)

.993

REDNESS

177: 0.993

. . .

235: 0.325

. . .

ROUNDNESS

235: 0.999

. . .

177: 0.406

. . .

Overall score for 177: min(0.993, 0.406) = .406

Overall score for 235: min(0.325, 0.999) = .325

Threshold value: min(0.993, 0.999) = .993

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999
139: 0.991	666: 0.996
. . .	. . .
235: 0.325	177: 0.406
. . .	. . .

Threshold Algorithm: Example (using min)

.991

REDNESS
177: 0.993
139: 0.991
. . .
235: 0.325
. . .

ROUNDNESS
235: 0.999
666: 0.996
. . .
177: 0.406
. . .

Threshold value: min(0.991, 0.996) = .991

Threshold Algorithm: Example (using min)

REDNESS	ROUNDNESS
177: 0.993	235: 0.999
139: 0.991	666: 0.996
702: 0.982	820: 0.992
. . .	. . .
235: 0.325	177: 0.406
. . .	. . .

Threshold Algorithm: Example (using min)

REDNESS
177: 0.993
139: 0.991
702: 0.982
. . .
235: 0.325
. . .

ROUNDNESS
235: 0.999
666: 0.996
820: 0.992
. . .
177: 0.406
. . .

.982

Threshold value: min(0.982 , 0.992) = .982

Properties of TA

- Correctness: For each monotone f and each database D of objects, **TA** finds the top k objects.
- Ease of implementation: Requires only **bounded** buffers
- Robustness: easy to extend to approximate top-k and stopping with guarantee
- No independence assumption needed

Correctness of the Halting Rule

Suppose the current top k objects have scores at least T (the current threshold).

Assume (by way of contradiction):

R unseen; S in current top k ; $\boxed{f(R)>f(S)}$

R has scores $x_1, \ldots, x_m$

$\Rightarrow x_i \leq t_i$ for every i (as R has not been seen)

$\Rightarrow \boxed{f(R) = f(x_1,\ldots, x_m) \leq f(t_1,\ldots, t_m) = T \leq f(S)}$

$\Rightarrow$ **contradiction!**

TA vs. FA

Proposition: TA halts at least as early as FA halts.

Proof: When FA halts, each of the *k* objects that appear in all lists have overall score at least as big as the current threshold, by monotonicity.

Example where TA beats FA (using min, *k*=1)

REDNESS
1: 0.9
3: 0.6
. . .
2: 0.2
4: 0.1

ROUNDNESS
2: 0.9
4: 0.8
. . .
1: 0.7
3: 0.1

Overall score for 1: min(0.9, 0.7) = 0.7

Threshold = min(0.6,.0.8) = 0.6

TA halts

FA has not seen an object in both lists, so does not halt

Instance Optimality

$\mathbf{A}$ = class of algorithms,

$\mathbf{D}$ = class of legal inputs.

For $A \in \mathbf{A}$ and $D \in \mathbf{D}$ have $cost(A,D) \geq 0$.

- An algorithm $A \in \mathbf{A}$ is **instance optimal** over $\mathbf{A}$ and $\mathbf{D}$ if there are constants c_1 and c_2 s.t.

  for every $A' \in \mathbf{A}$ and $D \in \mathbf{D}$

$$cost(A,D) \leq c_1 \cdot cost(A',D) + c_2.$$

c_1 is called the **optimality ratio**

Instance Optimality of TA

Intuition about why TA is instance optimal: Cannot stop **any sooner**, since the **next** object to be explored might have the **threshold value**.

But, life is a bit more delicate...

Wild Guesses

Wild guesses: random access for a field i of object R that has not been sequentially accessed before

- Neither FA nor TA use wild guesses
- Subsystem might not allow wild guesses

Instance Optimality- No Wild Guesses

Theorem: For each monotone f let

- $\mathbf{A}$ be the class of algorithms that
  - ***correctly*** find top k answers, with scoring function f, for ***every*** database.
  - Do not make wild guesses.
- $\mathbf{D}$ be the class of all databases.

Then **TA** is instance optimal over $\mathbf{A}$ and $\mathbf{D}$.

Optimality ratio is $m+m^2 \cdot c_R/c_S$ - best possible!

Our "threshold algorithm" is an even better algorithm (optimal in a stronger sense)

Amnon Lotem

Moni Naor

Ron Fagin

Laura Haas

But Ron, you told me that your algorithm is optimal!?

Well, Laura, there is optimal, and then there is optimal

Rank Aggregation vs. Score Aggregation

- Rank aggregation: Given sorted lists (permutations) $L_1, L_2, \ldots, L_m$ to be aggregated, Kemeny's criterion says that the consensus list is one where the sum of the distances to the L_i 's is minimal.
  - Using the Kendall τ distance (suggested by Kemeny) gives NP-hard optimization problem
- Score aggregation was considered trivial
  - Simple, efficient algorithm
  - Our new twist is to minimize the number of accesses

Influence

- We submitted the paper to PODS '01
- I was worried that the Threshold Algorithm was so simple that the paper would be rejected
  - So I called it a "remarkably simple algorithm"
  - The paper won the PODS Best Paper Award!
- The paper was very influential
  - Over 1800 citations (Google Scholar)
  - PODS Test of Time Award in 2011
  - IEEE Technical Achievement Award in 2011
  - Gödel Prize in 2014

Thanks to Mike Franklin

- Removed himself from the paper, since he was on the PODS ‘01 PC

Applications of TA

- relational databases
- multimedia databases
- music databases
- semistructured databases
- text databases
- uncertain databases
- probabilistic databases
- graph databases
- spatial databases
- spatio-temporal databases
- web-accessible databases
- XML data
- web text data
- semantic web
- high-dimensional datasets
- information retrieval
- fuzzy data sets
- data streams
- search auctions
- wireless sensor networks
- distributed sensor networks
- distributed networks
- social-tagging networks
- document tagging systems
- peer-to-peer systems
- recommender systems
- personal information management systems
- group recommendation systems
- document annotation