Optimal aggregation algorithms for middleware

CS856 Fall 2005 Presentation

Weihan Wang
w23wang@uwaterloo.ca

November 23, 2005

About the paper

- Ronald Fagin, IBM Research
- Amnon Lotem, Maryland
- Moni Naor, Weizmann, Israel


Agenda

- Background
- Fagin’s algorithm
- Threshold algorithm
- 8-approximation
- NRA algorithm
- Combined algorithm
Agenda

- Background
  - Fagin's algorithm
  - Threshold algorithm
  - \( \theta \)-approximation
  - NRA algorithm
  - Combined algorithm

Motivation: top-\( k \) queries

Find a girl with long hair, brown eyes, and sweet voice

Top-1 query with 3 attributes

Motivation: top-\( k \) queries
- Multimedia DB: “find 10 pictures that are funny and large in size”
- Info. retrieval: “find 100 papers that are most relevant to my research areas”
- Data stream: “find 5 users with the largest bandwidth usage”

- Live examples:
  - QBIC: www.qbic.almaden.ibm.com
  - Flickr: www.flickr.com
  - WinFS for Windows Vista
### WinFS for Windows Vista

![WinFS for Windows Vista](image)

### Data model

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$L_1$</th>
<th>$L_2$</th>
<th>$L_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.1</td>
<td>.2</td>
<td>.7</td>
<td>E</td>
<td>.8</td>
</tr>
<tr>
<td>B</td>
<td>.5</td>
<td>.3</td>
<td>.3</td>
<td>B</td>
<td>.5</td>
</tr>
<tr>
<td>C</td>
<td>.1</td>
<td>.9</td>
<td>.4</td>
<td>D</td>
<td>.3</td>
</tr>
<tr>
<td>D</td>
<td>.3</td>
<td>.1</td>
<td>.9</td>
<td>C</td>
<td>.1</td>
</tr>
<tr>
<td>E</td>
<td>.8</td>
<td>.4</td>
<td>.6</td>
<td>A</td>
<td>.1</td>
</tr>
</tbody>
</table>

### Data integration sys. (middleware)

![Data integration sys. (middleware)](image)
Problem definition

Middleware

Top-k results

Aggregation function: $t(x_1, x_2, x_3, x_4)$

Sort by attributes

$L_1$  $L_2$  $L_3$  $L_4$

Aggregation functions

- Can be max(), min(), avg(), ...

Aggregation functions

Sorted and random access

Middleware

sorted

$L_i$

1  2  3  4

random

......

$N$
Agenda

- Background
- Fagin’s algorithm
- Threshold algorithm
- B-approximation
- NRA algorithm
- Combined algorithm

---

Fagin’s algorithm (FA)

Suppose $m=3$, $k=2$; objects are A, B, C, ..., Z

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>NEUCQAYKTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$</td>
<td>JXCSBICMDO</td>
</tr>
<tr>
<td>$L_3$</td>
<td>CHFZDVFMBRX</td>
</tr>
</tbody>
</table>

Stop when there are $k$ objects, such that each of them has been seen in each list.

---

Fagin’s algorithm (FA): step 2

- For each object $R$ has been seen:
  - Do random access to get all of its attributes.
  - Calculate $t(R)$.
- Sort all these objects and output the first $k$ objects.

FA is correct, but not always optimal.
Agenda

1. Background
2. Fagin’s algorithm
3. Threshold algorithm
4. \(\theta\)-approximation
5. NRA algorithm
6. Combined algorithm

Threshold algorithm (TA)

\(m=3, k=2\)

\(\begin{align*}
L_1 & \rightarrow N \\
L_2 & \rightarrow J \\
L_3 & \rightarrow N
\end{align*}\)

\(t(J) > t(C) > t(N)\)

Output set \(Y\)

Threshold algorithm (TA)

\(m=3, k=2\)

\(\begin{align*}
L_1 & \rightarrow J \\
L_2 & \rightarrow X \\
L_3 & \rightarrow H
\end{align*}\)

\(t(H) > t(J)\)

Output set \(Y\)
Threshold algorithm (TA)

\( m=3, k=2 \)

\[ \begin{align*}
L_1 & \text{ NEUSCAVKE } \\
L_2 & \text{ JXKGSBICW } \\
L_3 & \text{ CHPZDVMPF } \\
\end{align*} \]

Output set \( Y \)

\( t(D) \)

Stop when the grade of the last object in \( Y \) is equal or larger than the threshold value.

Threshold algorithm (TA)

\( m=3, k=2 \)

\[ \begin{align*}
L_1 & \text{ NEUSCAVKE } \\
L_2 & \text{ JXKGSBICW } \\
L_3 & \text{ CHPZDVMPF } \\
\end{align*} \]

Threshold value \( \tau = t(D_x, t_y, t_z) \)

Output set \( Y \)

Middleware cost

- In this paper, we use middleware cost to measure optimality of an algorithm.
- To answer a query on database \( D \), an algorithm \( A \) needs:
  - \( s \) sorted accesses
  - \( r \) random accesses
- The middleware cost of \( A \) on \( D \) is:

\[ \text{cost}(A,D) = sc_s + rc_r \]
Instance optimality

- A set of databases: \(D\)
- A set of (middleware) algorithms: \(A\)
- \(B \in A\) is instance optimality if:
  \[
  \text{cost}(B, D) \leq c \cdot \text{cost}(A, D) + c'
  \]
  for every \(A \in A\) and \(D \in D\)
- \(c\): optimality ratio

Instance optimality of TA

- Assumptions
  - \(t()\): monotone
  - \(D\): all
  - \(A\): no wild guess
- Optimality: TA is instance optimal, with optimality ratio \(m + m(m-1)c_R/c_s\)

Instance optimality of TA (2)

- Assumptions
  - \(t()\): strictly monotone
  - \(D\): unique
  - \(A\): all
- Optimality: TA is instance optimal
Agenda

- Background
- Fagin’s algorithm
- Threshold algorithm
- $\theta$-approximation
- NRA algorithm
- Combined algorithm

$\theta$-approximation

$Y_k$ is the last object in $Y$, $\theta = \tau / t(Y_k)$

Guarantee: $\theta t(Y) \geq t(z)$

Instance optimality of $\theta$-approximation

- Assumptions
  - $t()$: monotone
  - $D$: all
  - $A$: no wild guess
  - $\theta > 1$
- Optimality: $\theta$-approximation is instance optimal
Agenda

1. Background
2. Fagin’s algorithm
3. Threshold algorithm
4. \( \theta \)-approximation
5. No-Random-Access algorithm
6. Combined algorithm

Lower/upper bound of an object

- Define Lower bound \( LB() \) as the value of \( t() \) when setting all unknown attributes to 0.
- Define Upper bound \( UB() \) as the value of \( t() \) when setting all unknown attributes to \( x \).

\[
\begin{align*}
L_1 & \quad 0.5 \\
L_2 & \quad 0.2 \\
L_3 & \quad 0.3 \\
L_4 & \quad 0.8 \\
\end{align*}
\]

\[
\begin{align*}
LB(A) &= t(0, 0, 0.8) \\
UB(A) &= t(0.5, 0.2, 0.8)
\end{align*}
\]

NRA algorithm

\[
\begin{align*}
\text{t}(\cdot) & \quad \text{UB} \\
\text{LB} & \quad \text{Output set } Y \\
& \quad \text{Other seen objects}
\end{align*}
\]
NRA algorithm

\[ t() \]

Output set \( Y \)  Other seen objects

Instance optimality of NRA

- Assumptions
  - \( t() \): monotone
  - \( D \): all
  - \( A \): no random access
- Optimality: NRA is instance optimal

Agenda

- Background
- Fagin’s algorithm
- Threshold algorithm
- \( \theta \)-approximation
- NRA algorithm
- Combined algorithm
Combined algorithm (CA)

For every $c_2/c_4$ step, obtain all unknown attr. of the object with the largest UB.

Instance optimality of CA

- Assumptions
  - $t()$: strictly monotone in each argument
  - $D$: unique
  - $A$: all

- Optimality: CA is instance optimal

Agenda

- Background
- Fagin’s algorithm
- Threshold algorithm
- $\theta$-approximation
- NRA algorithm
- Combined algorithm
Conclusion

- TA is instance optimal in most cases
- $\theta$-approx: early stop
- NRA: random access is not allowed
- CA: random access is costly

Future work
- Tightly instance optimal
- More efficient structure of NRA
- Compare CA vs. TA

Discussion

- Object caching of TA
- Grades output of NRA
- Other metrics for algorithm optimality
- Assumptions on databases
I.O. in other fields

- Competitive analysis
- Approximation algorithms
- The mean of Monte Carlo estimation (Dagum et al.)
- Operations on sorted sets (Demaine et al.)

Memory overhead

- FA: need to remember $t()$ for all objects that have been seen.
- TA: only need to remember $t()$ for objects in $Y$.
- NRA: similar to FA.

Wild guess example

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>(2$n$ + 1, 1)</td>
</tr>
<tr>
<td>(2, 1)</td>
<td>(2$n$, 1)</td>
</tr>
<tr>
<td>(3, 1)</td>
<td>(2$n$ − 1, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(n + 1, 1)</td>
<td>(n + 1, 1)</td>
</tr>
<tr>
<td>(n + 2, 0)</td>
<td>(n, 0)</td>
</tr>
<tr>
<td>(n + 3, 0)</td>
<td>(n − 1, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(2$n$ + 1, 0)</td>
<td>(1, 0)</td>
</tr>
</tbody>
</table>
Instance Optimality w/ wild guess

- **Assumptions**
  - $t()$: $\min(x_1, x_2)$
  - $D$: all
  - $A$: all
- **Optimality**: no algorithm is instance optimal

Wild guess example w/ $\theta$

<table>
<thead>
<tr>
<th>$L_1$</th>
<th>$L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1, \cdot)$</td>
<td>$(2n + 1, \cdot)$</td>
</tr>
<tr>
<td>$(2, \cdot)$</td>
<td>$(2n, \cdot)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$(n + 1, \frac{1}{2})$</td>
<td>$(n + 1, \frac{1}{2})$</td>
</tr>
<tr>
<td>$(n + 2, \frac{1}{2^{2n}})$</td>
<td>$(n, \frac{1}{2^{2n}})$</td>
</tr>
<tr>
<td>$(n + 3, \cdot)$</td>
<td>$(n - 1, \cdot)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$(2n + 1, \cdot)$</td>
<td>$(1, \cdot)$</td>
</tr>
</tbody>
</table>
Costs for top-1 & top-2 for NRA

<table>
<thead>
<tr>
<th>( L_1 )</th>
<th>( L_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R, 1)</td>
<td>(R, 0)</td>
</tr>
<tr>
<td>(( \frac{1}{2} ))</td>
<td>(( \frac{1}{2} ))</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(( \frac{1}{2} ))</td>
<td></td>
</tr>
</tbody>
</table>

Assumptions
- \( t() \): min()
- \( D \): unique
- \( A \): all

Optimality: CA is instance optimal

Instance optimality of CA (2)

Instance optimality dependency

Assumptions
- \( t() \): min()
- \( D \): all
- \( A \): no wild guess

Optimality: no algorithm has instance optimality independent of \( C_R/C_S \)