Tell Me Who I Am: An Interactive Recommendation System

N. Alon, B. Awerbuch, Y. Azar, B. Patt-Shamir

Richard Huber
Publication

- Theory of Computing Systems
  - Volume 45
  - August 2009

- Tel Aviv University, Israel
- John Hopkins University, Baltimore, USA
Experiment

- Travel in a foreign country
- Unknown language
- Learn to know the night life subculture
- Not allowed to talk to each other
Experiment

- Problem:
  - 5 typical drinks
  - money for 3 drinks
- Waitress asks whether you liked the drink
- Idea: Human preferences correlate
Experiment

http://demo.racerfish.com
Players and Billboard

How can a player find out his preferences with only a few probes?
Statement of the Problem

- n players and m objects
- each player has an unknown yes/no grade for each object
- Parallel rounds: in each round each player
  - reads the shared billboard
  - probes one object
  - writes the result of the probe on the billboard
- For each player: output a vector as close as possible to that player's original preference vector
Statement of the Problem (Formal)

- **Input:**
  - A set $P$ of $n$ players and a set $O$ of $m$ objects
  - A vector $v(p) \in \{yes, no\}^m$ for each player $p$

- **Output:**
  - An estimate vector $w(p) \in \{yes, no\}^m$ for each player $p$

- **Goal:**
  - Minimize $dist(v(p), w(p))$ for each player $p$
  - $dist(x, y)$ is the Hamming distance
  - Minimize the number of probes
Input Characteristic

- **Diameter** of a subset $A \subseteq P$
  \[
  D(A) = \max \{ \text{dist}(v(p), v(q)) | p, q \in A \}
  \]

- **$(\alpha, D)$-typical set**: Subset $A \subseteq P$ with
  \[
  |A| \geq \alpha n, \quad 0 \leq \alpha \leq 1
  \]
  \[
  D(A) \leq D, \quad D \geq 0
  \]
Approximation Quality

- **Discrepancy** of a subset $A \subseteq P$
  \[
  \Delta(A) = \max \{ \text{dist}(w(p), v(p)) | p \in A \}
  \]

- **Stretch** of a subset $A \subseteq P$
  \[
  \rho(A) = \frac{\Delta(A)}{D(A)}
  \]
The CHOOSE_CLOSEST Problem

- **Input**
  - A set $V$ of preference Vectors with $|V| = k$
  - A player $p$ with (initially unknown) preference vector $v(p)$

- **Output**
  - A vector $w_{min} \in V$ such that
    $$\text{dist}(w_{min}, v(p)) \leq \text{dist}(w, v(p)), w \in V$$

<table>
<thead>
<tr>
<th>Player p</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_1</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>v_2</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>v_3</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
The SELECT Algorithm

- Solves an adapted version of the CHOOSE_CLOSEST problem
- Adapts:
  - Additional input $D$
  - There is a vector $w \in V$ such that $\text{dist}(w, v(p)) \leq D$
The SELECT Algorithm

\[
\begin{array}{|c|cccccccc|}
\hline
D=1 & \text{Object 1} & \text{Object 2} & \text{Object 3} & \text{Object 4} & \text{Object 5} & \text{Object 6} & \text{Object 7} \\
\hline
\hline
\text{V} & v_1 & \text{yes} & \text{no} & \text{yes} & \text{no} & \text{no} & \text{yes} & \text{yes} \\
\hline
v_2 & \text{yes} & \text{no} & \text{no} & \text{yes} & \text{yes} & \text{no} & \text{no} \\
\hline
v_3 & \text{yes} & \text{yes} & \text{no} & \text{yes} & \text{yes} & \text{no} & \text{no} \\
\hline
\end{array}
\]

1) Repeat
   1a) Let \( X(V) \) be the set of Objects on which some two vectors in \( V \) differ.
   1b) Execute Probe on the first coordinate in \( X(V) \) that has not been probed yet.
   1c) Remove from \( V \) any vector with more than \( D \) disagreements with \( v(p) \).

Until all coordinates in \( X(V) \) are probed or \( X(V) \) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object 1</td>
</tr>
<tr>
<td>(v_1)</td>
<td>yes</td>
</tr>
<tr>
<td>(v_2)</td>
<td>yes</td>
</tr>
<tr>
<td>(v_3)</td>
<td>yes</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let \(X(V)\) be the set of Objects on which some two vectors in \(V\) differ.
   1b) Execute Probe on the first coordinate in \(X(V)\) that has not been probed yet.
   1c) Remove from \(V\) any vector with more than \(D\) disagreements with \(v(p)\).

Until all coordinates in \(X(V)\) are probed or \(X(V)\) is empty.
**The SELECT Algorithm**

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object 1</td>
</tr>
<tr>
<td>V</td>
<td>?</td>
</tr>
<tr>
<td>v_1</td>
<td>yes</td>
</tr>
<tr>
<td>v_2</td>
<td>yes</td>
</tr>
<tr>
<td>v_3</td>
<td>yes</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let X(V) be the set of Objects on which some two vectors in V differ.
   1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
   1c) Remove from V any vector with more than D disagreements with v(p).
Until all coordinates in X(V) are probed or X(V) is empty.
The **SELECT Algorithm**

<table>
<thead>
<tr>
<th>D=1</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V$_1$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>V$_2$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>V$_3$</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let X(V) be the set of Objects on which some two vectors in V differ.
   1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
   1c) Remove from V any vector with more than D disagreements with v(p). Until all coordinates in X(V) are probed or X(V) is empty.
## The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>v_1</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>v_2</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>v_3</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let $X(V)$ be the set of Objects on which some two vectors in $V$ differ.
   1b) Execute Probe on the first coordinate in $X(V)$ that has not been probed yet.
   1c) Remove from $V$ any vector with more than $D$ disagreements with $v(p)$.

Until all coordinates in $X(V)$ are probed or $X(V)$ is empty.
### The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object 1</td>
</tr>
<tr>
<td>Player p</td>
<td>?</td>
</tr>
<tr>
<td>V</td>
<td>v_1</td>
</tr>
<tr>
<td></td>
<td>v_2</td>
</tr>
<tr>
<td></td>
<td>v_3</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let X(V) be the set of Objects on which some two vectors in V differ.
   1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
   1c) Remove from V any vector with more than D disagreements with v(p).

Until all coordinates in X(V) are probed or X(V) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player p</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( v_1 )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( v_3 )</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let \( X(V) \) be the set of Objects on which some two vectors in \( V \) differ.
   1b) Execute Probe on the first coordinate in \( X(V) \) that has not been probed yet.
   1c) Remove from \( V \) any vector with more than \( D \) disagreements with \( v(p) \).

Until all coordinates in \( X(V) \) are probed or \( X(V) \) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object 1</td>
</tr>
<tr>
<td>Player p</td>
<td>?</td>
</tr>
<tr>
<td>V</td>
<td>v₁</td>
</tr>
<tr>
<td></td>
<td>v₂</td>
</tr>
<tr>
<td></td>
<td>v₃</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let X(V) be the set of Objects on which some two vectors in V differ.
   1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
   1c) Remove from V any vector with more than D disagreements with v(p).
Until all coordinates in X(V) are probed or X(V) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object 1</td>
</tr>
<tr>
<td>Player p</td>
<td>?</td>
</tr>
<tr>
<td>V</td>
<td>v₁</td>
</tr>
<tr>
<td></td>
<td>v₂</td>
</tr>
<tr>
<td></td>
<td>v₃</td>
</tr>
</tbody>
</table>

1) Repeat
   1a) Let X(V) be the set of Objects on which some two vectors in V differ.
   1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
   1c) Remove from V any vector with more than D disagreements with v(p).

Until all coordinates in X(V) are probed or X(V) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th></th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=1</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td><strong>v1</strong></td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td><strong>v2</strong></td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td><strong>v3</strong></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

1) Repeat

1a) Let X(V) be the set of Objects on which some two vectors in V differ.
1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
1c) Remove from V any vector with more than D disagreements with v(p).

Until all coordinates in X(V) are probed or X(V) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>X(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1</td>
<td>Object 2</td>
</tr>
<tr>
<td>Player p</td>
<td>?</td>
</tr>
<tr>
<td>v₁</td>
<td>yes</td>
</tr>
<tr>
<td>v₂</td>
<td>yes</td>
</tr>
<tr>
<td>v₃</td>
<td>yes</td>
</tr>
</tbody>
</table>

1) Repeat

1a) Let X(V) be the set of Objects on which some two vectors in V differ.
1b) Execute Probe on the first coordinate in X(V) that has not been probed yet.
1c) Remove from V any vector with more than D disagreements with v(p).
Until all coordinates in X(V) are probed or X(V) is empty.
The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player p</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( v_1 )</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>( v_3 )</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

2) **Let Y be the set of objects probed by p.** Output the vector closest to \( v(p) \) regarding only the objects in Y.
## The SELECT Algorithm

<table>
<thead>
<tr>
<th>D=1</th>
<th>Object 1</th>
<th>Object 2</th>
<th>Object 3</th>
<th>Object 4</th>
<th>Object 5</th>
<th>Object 6</th>
<th>Object 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player p</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>V</td>
<td>v₁</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>v₂</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>v₃</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

2) Let Y be the set of objects probed by p. **Output the vector closest to v(p) regarding only the objects in Y.**
The SELECT Algorithm: Correctness

- Any vector removed from V is at distance more than D from v(p).
- All distinguishing coordinates of the remaining vectors were probed.
- Distance to v(p) exactly known up to a common additive term.
The SELECT Algorithm: Cost

- Each probe exposes at least one disagreement.
- No vector remains in V after finding D+1 disagreements.
- After k(D+1) probes, no vector remains in V (k is the number of Vectors in V).
- Total cost upper bounded by k(D+1).
The ZERO_RADIUS Algorithm

- **Input:**
  - A set of players $P$ and a set of objects $O$
  - Parameter $\alpha, \ 0 \leq \alpha \leq 1$

- **Output:**
  - The correct vector for all players in a $(\alpha, 0)$-typical set

- Fails with probability $n^{-\Omega(1)}$

- Terminates after $\mathcal{O}\left(\frac{\log(n)}{\alpha}\right)$ probes
The ZERO_RADIUS Algorithm

1) If \( \min(|P|, |O|) \leq \frac{c \ln n}{\alpha} \) probe all objects and return
The ZERO_RADIUS Algorithm

2) Partition randomly $P = P_1 \cup P_2$ and $O = O_1 \cup O_2$
The ZERO_RADIUS Algorithm

3) Recursively execute ZERO_RADIUS for the yellow areas
The ZERO_RADIUS Algorithm

3) Recursively execute ZERO_RADIUS for the yellow areas
The ZERO_RADIUS Algorithm

3) Recursively execute ZERO_RADIUS for the yellow areas
The ZERO_RADIUS Algorithm

4) Consider only vectors, which are returned by a $\alpha/2$ fraction of the players.
The ZERO_RADIUS Algorithm

5) Execute SELECT for the green areas with the $\alpha/2$ remaining orange vectors as input and D=0
ZERO_RADIUS: Cost Analysis

- **Step 1) Probing whole sub-area**
  - Executed at most once by each player
  - How many objects probed by each player?
    - Recursive halving maintains $|O| \approx |P| \cdot m/n$
    - $n < m$:
      - Recursion stops when $|P| = O'(\log n/\alpha)$
      - Player probes $O'(m/n \cdot \log n/\alpha)$ objects
    - $n \geq m$:
      - Recursion stops when $|O| = O'(\log n/\alpha)$
      - Player probes $O'(\log n/\alpha)$ objects
  - Total cost of step 1) per player is $O'([m/n] \log n/\alpha)$
Zero_Radius: Cost Analysis

- Step 5) (call to SELECT)
  - Call SELECT with $O\big(\frac{1}{\alpha}\big)$ candidates and $D=0$
  - Recursion depth upper bounded by $O\big(\log n\big)$
  - Total cost per player upper bounded by $O\big(\log n / \alpha\big)$

- **Zero_Radius** terminates after

$$O\left(\left\lceil \frac{m}{n} \right\rceil \log n \cdot \frac{1}{\alpha}\right) + O\left(\frac{\log n}{\alpha}\right) = O\left(\left\lceil \frac{m}{n} \right\rceil \frac{\log n}{\alpha}\right)$$

probes
Summary

- **SELECT**
  - Find closest of $k$ vectors within distance $D$
  - $k(D+1)$

- **ZERO_RADIUS**
  - Find correct preference vector for players in $(\alpha, 0)$-typical sets
  - $O\left(\lceil m/n \rceil \log n/\alpha \right)$
The SMALL_RADIUS Algorithm

- **Input**
  - Parameter $\alpha$, $0 \leq \alpha \leq 1$
  - Parameter $D = O(\log n)$

- **Output**
  - An estimate vector $w(p)$ for every player $p$ which is a member of a $(\alpha, D)$-typical set $A$ with
    \[
    \text{dist}(w(p), v(p)) \leq 5D, \quad p \in A
    \]
    \[
    \Rightarrow \Delta(A) \leq 5D
    \]
    \[
    \Rightarrow \rho(A) \leq 5
    \]
The SMALL_RADIUS Algorithm

1) Partition randomly \( O = O_1 \cup \ldots \cup O_s \) with \( s = D^{3/2} \)
The SMALL_RADIUS Algorithm

2) For every $O_i$, execute ZERO_RADIUS with all players and parameter $\alpha/5$
The SMALL_RADIUS Algorithm

3) Within the set $O_i$, only use vectors output by at least $\alpha n/5$ players
The SMALL_RADIUS Algorithm

4) Within the set $O_i$, player P applies procedure SELECT to the remaining vectors with distance bound $D$
The SMALL_RADIUS Algorithm

5) Do this $K$ times.

Probability that one of the $K$ independent executions succeed is $1 - 2^{-\Omega(K)}$
The SMALL_RADIUS Algorithm

6) On the successful executions, all players execute SELECT with distance bound 5D and output the result.
SMALL_RADIUS: Cost

- Step 2): ZERO_RADIUS invoked
  - $s = \mathcal{O}(D^{3/2})$ times with $n$ users and $m/s$ objects
    $$\mathcal{O}\left(\left(\frac{m}{n} + D^{3/2}\right) \cdot \frac{\log n}{\alpha}\right)$$

- Step 4): SELECT invoked
  - $s = \mathcal{O}(D^{3/2})$ times with bound $D$ and at most $\mathcal{O}(1/\alpha)$ candidates
    $$\mathcal{O}(D^{5/2}/\alpha)$$

- Step 6): SELECT invoked $\mathcal{O}(KD)$

- Overall complexity
  $$\mathcal{O}\left(K \frac{m}{\alpha n} D^{3/2} (\log n + D)\right)$$
Summary

- **SELECT**
  - Find closest of $k$ vectors within distance $D$
  - $k(D+1)$

- **ZERO_RADIUS**
  - Find correct preference vector for players in $(\alpha, 0)$-typical sets
  - $O\left(\lceil m/n \rceil \log n/\alpha \right)$

- **SMALL_RADIUS**
  - Find preference vectors of $(\alpha, D)$-typical sets with $\rho \leq 5$
  - $O\left(\frac{m}{\alpha n} D^{3/2} \left( \log n + D \right) \right)$
The LARGE_RADIUS Algorithm

- **Input**
  - Parameter $\alpha$
  - Parameter $D \geq \Omega(\log n)$

- **Output**
  - An estimate vector $w(p)$ for every player $p$ which is a member of a $(\alpha, D)$-typical set $A$ with
    
    $$
    \text{dist}(w(p), v(p)) = O\left(\frac{D}{\alpha}\right), \quad p \in A
    $$
    
    $$
    \Rightarrow \Delta(A) = O\left(\frac{D}{\alpha}\right)
    $$
    
    $$
    \Rightarrow \rho(A) = O\left(\frac{1}{\alpha}\right)
    $$
LARGE_RADIUS: Idea
Main Algorithm

- Given $\alpha$ and $D$
  - If $D = 0$ use ZERO_RADIUS
  - If $D = O(\log n)$ use SMALL_RADIUS
  - If $D \geq \Omega(\log n)$ use LARGE_RADIUS

- For every $(\alpha, D)$-typical set $A$
  - w.h.p. $\Delta(A) = O(D/\alpha)$
  - the number of probes performed by each player is
    \[ O\left(\left\lceil \frac{m}{n} \right\rceil \cdot \frac{\log^{7/2} n}{\alpha^2} \right) \]
Unknown Input Characteristics

- Known $\alpha$, unknown $D$
  - Run $O(\log n)$ independent versions of the main algorithm with $D = \{0, 2^1, 2^2, \ldots, 2^{\log n}\}$
  - Choose closest of all $O(\log n)$ output vectors
  - Increase running time by a factor of $O(\log n)$
  - Decrease quality of output by a constant factor

$$O\left(\left\lceil \frac{m}{n} \right\rceil \cdot \frac{\log^{9/2} n}{\alpha^2} \right)$$
Unknown Input Characteristics

- Unknown $\alpha$, unknown $D$
  - Given $\alpha$ => number of probing rounds $\tau = O\left(\left\lfloor \frac{m}{n} \right\rfloor \cdot \frac{\log^{9/2} n}{\alpha^2} \right)$
  - Given $\tau$ => minimum $\alpha(\tau)$
  - Start parallel versions with $\alpha(\tau = 2^j)$ and unknown $D$
  - After every round, choose closest output vector
Conclusion

- Distributed algorithm for an interactive recommendation system
  - No restrictions on the input set
  - Has polylogarithmic running time
- First algorithm published that combines these two properties