Aggregating Inconsistent Information: Ranking and Clustering

Nir Ailon
Moses Charikar
Alantha Newman
Rank Aggregation

- Finite space $V$ of $n$ objects (candidates).
- “Democratically” choose ranking of $V$.
- Input: Voters’ rankings (permutations) $\pi_1, \ldots, \pi_k$ on $V$. Output: “Aggregate” ranking $\pi$.
- Condorcet’s Paradox:
  \[ \pi_1 : A < B < C \quad \pi_2 : B < C < A \quad \pi_3 : C < A < B \]
  Majority ranks $A$ before $B$
  Majority ranks $B$ before $C$
  Majority ranks $C$ before $A$
Rank Aggregation: Minimizing the Kendall-tau Distance

• Find $\pi$ minimizing

$$\sum_{i=1}^{k} \text{dist}(\pi, \pi_i) .$$

• Kendall tau distance function:

$$\text{dist}(\pi, \sigma) = \# \{u, v \in V | u <_{\pi} v \text{ and } u >_{\sigma} v\} .$$

• Satisfies desirable Condorcet property
Related Graph Problem:
Minimum Feedback Arc Set (FAS) in Tournament

• Given tournament $G = (V, A)$:
  \[ \forall u \neq v : (u, v) \in A \text{ or } (v, u) \in A. \]

• Find permutation $\pi$ minimizing:
  \[ \# \{ u, v \mid (u, v) \in A \text{ and } v <_\pi u \}. \]
  (Number of backward edges)
Weighted FAS in Tournament

- Given weighted tournament $G = (V, w)$:
  \[ \forall u \neq v : w_{uv} \geq 0 . \]

- Find permutation $\pi$ minimizing:
  \[ \sum_{u < \pi v} w_{vu} . \]

- **Probability constraints**: $w_{uv} + w_{vu} = 1 .

- **Triangle inequality constraints**: $w_{uv} \leq w_{uy} + w_{yv} .


Rank Aggregation as Weighted FAS in Tournament

- Set $w_{uv}$ as fraction voters ranking $u$ before $v$.
- Satisfies
  - Probability constraints.
  - Triangle inequality constraints.
    For $k = 1$ voter: equivalent to having no 3-cycles.
    For $k > 1$, by convex combination.

- Example:

<table>
<thead>
<tr>
<th></th>
<th>Winner</th>
<th>Runnerup</th>
<th>2nd Runnerup</th>
<th>Loser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter 1</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Voter 2</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Voter 3</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>
Rank Aggregation as Weighted FAS in Tournament

- Set $w_{uv}$ as fraction voters ranking $u$ before $v$.
- Satisfies
  - Probability constraints.
  - Triangle inequality constraints.
    - For $k = 1$ voter: equivalent to having no 3-cycles.
    - For $k > 1$, by convex combination.
- Example:

<table>
<thead>
<tr>
<th>Voter</th>
<th>Winner</th>
<th>Runnerup</th>
<th>2nd Runnerup</th>
<th>Loser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter 1</td>
<td>$A$</td>
<td>D</td>
<td>$B$</td>
<td>$C$</td>
</tr>
<tr>
<td>Voter 2</td>
<td>D</td>
<td>$B$</td>
<td>$C$</td>
<td>$A$</td>
</tr>
<tr>
<td>Voter 3</td>
<td>$C$</td>
<td>$B$</td>
<td>$A$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

$w_{AB} = 1/3$
Rank Aggregation as Weighted FAS in Tournament

- Set $w_{uv}$ as fraction voters ranking $u$ before $v$.
- Satisfies
  - Probability constraints.
  - Triangle inequality constraints.
    For $k = 1$ voter: equivalent to having no 3-cycles.
    For $k > 1$, by convex combination.
- Example:

<table>
<thead>
<tr>
<th>Voter 1</th>
<th>Winner</th>
<th>Runnerup</th>
<th>2nd Runnerup</th>
<th>Loser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter 2</td>
<td>D</td>
<td>$B$</td>
<td>$C$</td>
<td>$A$</td>
</tr>
<tr>
<td>Voter 3</td>
<td>C</td>
<td>$B$</td>
<td>$A$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

$w_{BA} = \frac{2}{3}$
Rank Aggregation as Weighted FAS in Tournament

- Set $w_{uv}$ as fraction voters ranking $u$ before $v$.

- Satisfies
  - Probability constraints.
  - Triangle inequality constraints.
    For $k = 1$ voter: equivalent to having no 3-cycles.
    For $k > 1$, by convex combination.

- Example:

<table>
<thead>
<tr>
<th>Voter 1</th>
<th>Winner</th>
<th>Runnerup</th>
<th>2nd Runnerup</th>
<th>Loser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter 2</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Voter 3</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

$w_{BC} = 2/3$
Rank Aggregation as Weighted FAS in Tournament

• Set $w_{uv}$ as fraction voters ranking $u$ before $v$.

• Satisfies
  
  – Probability constraints.
  – Triangle inequality constraints.

  For $k = 1$ voter: equivalent to having no 3-cycles.
  For $k > 1$, by convex combination.

• Example:

<table>
<thead>
<tr>
<th></th>
<th>Winner</th>
<th>Runnerup</th>
<th>2nd Runnerup</th>
<th>Loser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voter 1</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Voter 2</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Voter 3</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

$w_{CB} = 1/3$
Previous Results: Lower Bounds

- **Rank Aggregation** NP-hard even for \( k = 4 \) voters [DKNS01].
  (in P for \( k = 2 \) voters, unknown for \( k = 3 \).)

- Minimum FAS in general digraphs NP-hard to approximate to within factor 1.36 [DS02].

- Minimum FAS in tournaments was conjectured to be NP-hard [BJT92].
Previous Results: Upper Bounds

- **Rank Aggregation** approximable to within factor 2. (One of the $k$ voters’ ranking does it.)
  Other 2-approximation algorithms exist [DKNS01].

- Minimum FAS in general digraphs approximable to within factor $O(\log n \log \log n)$ [ENSS98, Sey95].
The 2-Approximation Algorithm

Given any metric $d(\cdot, \cdot)$ on $U$, subset $X \subseteq U$
The 2-Approximation Algorithm

Find $u_{opt} \in U$ minimizing $\sum_{x \in X} d(u, x)$.
The 2-Approximation Algorithm

Choosing random $u_{rand} \in X$ gives expected 2 approximation.
### Our Results (Combinatorial Algorithm)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3 (log $n \log \log n$)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Min FAS Tournament</strong></td>
<td>11/7 (2)</td>
</tr>
<tr>
<td><strong>Rank Aggregation</strong></td>
<td></td>
</tr>
</tbody>
</table>
Our Results (LP-Based Algorithm)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min FAS Tournament</td>
<td>2.5 (3)</td>
<td></td>
</tr>
<tr>
<td>Rank Aggregation</td>
<td>4/3 (11/7)</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm **KwikSort**

\[
\text{KwikSort}(G = (V, A))
\]

pick \( v \in V \) uniformly at random

let \( V_L = \{u \in V | (u, v) \in A\} \)
let \( G_L \) = subtournament induced by \( V_L \)

let \( V_R = \{u \in V | (v, u) \in A\} \)
let \( G_R \) = subtournament induced by \( V_R \)

return order \( \pi = [\text{KwikSort}(G_L), v, \text{KwikSort}(G_R)] \)
Algorithm **KwikSort**

**KwikSort**($G = (V, A)$)

pick $v \in V$ uniformly at random

let $V_L = \{ u \in V | (u, v) \in A \}$
let $G_L =$ subtournament induced by $V_L$

let $V_R = \{ u \in V | (v, u) \in A \}$
let $G_R =$ subtournament induced by $V_R$

return order $\pi = [\text{KwikSort}(G_L), v, \text{KwikSort}(G_R)]$

**Main Theorem:**
**KwikSort** is a randomized expected 3-approximation algorithm for minimum FAS in tournaments.
Proof of Main Theorem (1)

• How can an edge become a *backward edge*?

![Diagram]

• Charge backwardness of \((u, v)\) to directed triangle \((u, v, y)\).

• Let \(T\) be set of directed triangles.

• For \(t \in T\) define event \(A_t = \text{"t is charged"}\).

• Let \(p_t = \text{Pr}[A_t]\).

• 1-1 correspondence: backward edges \(\iff\) charged triangles.

\[\Rightarrow\] Expected cost of KwikSort: \(\sum_{t \in T} p_t\).
Proof of Main Theorem (2)

(A) Tournament on $V = \{u, v, y, y'\}$.
$T = \{t = (u, y, v), t' = (u, y', v)\}$.

(B) $y$ chosen as pivot.
Backwardness of $(u, v)$ charged to $t$.
Event $A_t$ occurred, $A_{t'}$ did not.

(C) $u$ chosen as pivot.
Backwardness of $(y, v)$ charged to $t$.
Backwardness of $(y', v)$ charged to $t'$.
Both $A_t, A_{t'}$ occurred.
Proof of Main Theorem (3)

• Extreme case: $T$ consists of edge-disjoint triangles.

$\Rightarrow |T|$ would be a lower bound of any solution.

• Also true fractionally:
  If $\{\beta_t \geq 0\}_{t \in T}$ is a fractional packing of $T$ (w.r.t. edges):
  \[
  \sum_{t \in T: e \in t} \beta_t \leq 1 \quad \forall e \in A ,
  \]
  then $\sum_{t \in T} \beta_t$ would be a lower bound of any solution.

• Can we pack triangles using $p_t$...?
Proof of Main Theorem (4)

- Equivalent definition of $A_t$:
  “All 3 vertices of $t$ are input to same recursive call to KWIKSORT when one of them was chosen as pivot”

- (Therefore edge $e$ not incident to pivot becomes backward)

- Let $B_e$=”Edge $e$ becomes backward”.

- For $e \in t$, $Pr[B_e|A_t] = 1/3$.

- Assume $e \in t, t'$. So $Pr[A_{t'}|A_t \land B_e] = 0$ (next slide...).

$\Rightarrow$ Events $(A_{t'} \land B_e)$ and $(A_t \land B_e)$ are disjoint.

$\Rightarrow$

$$\sum_{t \in T: e \in t} \frac{p_t}{3} \leq 1 \ \forall e \in A .$$

$\Rightarrow$ Lower bound for any solution: $\sum_{t \in T} \frac{p_t}{3}$.
\((A_t \land B_e)\) disjoint from \(A_{t'}, A_{t''}, A_{t'''}\)
Weighted Case

• For $e \in A_w$, let

$$\bar{w}_e = \text{cost if } e \text{ forward}$$

$$w_e = \text{cost if } e \text{ backward} \quad (\text{so } \bar{w}_e \leq w_e)$$

• For optimal solution $\pi^*$, let $c_e^*$ be cost of edge $e$:

$$c_e^* = \begin{cases} 
\bar{w}_e & \text{e forward w.r.t. } \pi^* \\
w_e & \text{e backward}
\end{cases}$$

• So $\text{cost}(\pi^*) = \sum_{e \in A_w} c_e^*$.

• Let $T$ be set of directed triangles in $G_w$.

• For $t \in T$, $t = \{e_1, e_2, e_3\}$, let

$$w(t) = w_{e_1} + w_{e_2} + w_{e_3}$$

$$c^*(t) = c_{e_1}^* + c_{e_2}^* + c_{e_3}^*$$
Main Lemma

If \( \exists \alpha \geq 1 \text{ s.t. } w(t) \leq \alpha c^*(t) \forall t \in T \), then \( E[\text{cost}(\pi)] \leq \alpha \text{cost}(\pi^*) \).

Proof: We charge cost of \( \pi \)-backward edges in \( G_w \) to triangles in \( T \).

• Expected cost charged to \( t \in T \) is

\[
\left( \frac{1}{3} p_t w_{e_1} + \frac{1}{3} p_t w_{e_2} + \frac{1}{3} p_t w_{e_3} \right) = \frac{1}{3} p_t w(t) .
\]

\[\Rightarrow\]

\[
E[\text{cost}(\pi)] = \sum_{t \in T} \frac{1}{3} p_t w(t) + \sum_{e \in A_w} \left( 1 - \sum_{t \in T: e \in t} \frac{1}{3} p_t \right) \bar{w}_e
\]

\[
\text{cost}(\pi^*) = \sum_{e \in A_w} c_e^* = \sum_{t \in T} \frac{1}{3} p_t c^*(t) + \sum_{e \in A_w} \left( 1 - \sum_{t \in T: e \in t} \frac{1}{3} p_t \right) c_e^*
\]
**Main Lemma**

If $\exists \alpha \geq 1$ s.t. $w(t) \leq \alpha c^*(t) \forall t \in T$, then $E[cost(\pi)] \leq \alpha cost(\pi^*)$.

**Proof:** We charge cost of $\pi$-backward edges in $G_w$ to triangles in $T$.

- **Expected cost charged to $t \in T$ is**
  
  $$
  \left(\frac{1}{3}p_tw_{e_1} + \frac{1}{3}p_tw_{e_2} + \frac{1}{3}p_tw_{e_3}\right) = \frac{1}{3}p_tw(t).
  $$

  $$
  \Rightarrow
  $$

  $$
  E[cost(\pi)] = \sum_{t \in T} \frac{1}{3}p_tw(t) + \sum_{e \in A_w} \left(1 - \sum_{t \in T:e \in t} \frac{1}{3}p_t\right)\bar{w}_e
  $$

  $$
  cost(\pi^*) = \sum_{e \in A_w} c^*_e = \sum_{t \in T} \frac{1}{3}p(tc^*(t)) + \sum_{e \in A_w} \left(1 - \sum_{t \in T:e \in t} \frac{1}{3}p_t\right)c_e^*
  $$

  $\Box$
Main Lemma

If $\exists \alpha \geq 1$ s.t. $w(t) \leq \alpha c^*(t) \forall t \in T$, then $E[\text{cost}(\pi)] \leq \alpha \text{cost}(\pi^*)$.

Proof: We charge cost of $\pi$-backward edges in $G_w$ to triangles in $T$.

- Expected cost charged to $t \in T$ is

$$
\left( \frac{1}{3} p_t w_{e_1} + \frac{1}{3} p_t w_{e_2} + \frac{1}{3} p_t w_{e_3} \right) = \frac{1}{3} p_t w(t).
$$

$\Rightarrow$

$$
E[\text{cost}(\pi)] = \sum_{t \in T} \frac{1}{3} p_t w(t) + \sum_{e \in A_w} \left( 1 - \sum_{t \in T : e \in t} \frac{1}{3} p_t \right) \bar{w}_e
$$

$$
\text{cost}(\pi^*) = \sum_{e \in A_w} c_e^* = \sum_{t \in T} \frac{1}{3} p_t c^*(t) + \sum_{e \in A_w} \left( 1 - \sum_{t \in T : e \in t} \frac{1}{3} p_t \right) c_e^*
$$
Main Lemma

If $\exists \alpha \geq 1$ s.t. $w(t) \leq \alpha c^*(t) \forall t \in T$, then $E[\text{cost}(\pi)] \leq \alpha \text{cost}(\pi^*)$.

Proof: We charge cost of $\pi$-backward edges in $G_w$ to triangles in $T$.

- Expected cost charged to $t \in T$ is

$$
\left( \frac{1}{3} p_t w_{e_1} + \frac{1}{3} p_t w_{e_2} + \frac{1}{3} p_t w_{e_3} \right) = \frac{1}{3} p_t w(t).
$$

$\Rightarrow$

$$
E[\text{cost}(\pi)] = \sum_{t \in T} \frac{1}{3} p_t w(t) + \sum_{e \in A} \left( 1 - \sum_{t \in T: e \in t} \frac{1}{3} p_t \right) \bar{w}_e
$$

$\text{cost}(\pi^*) = \sum_{e \in A} c_e^* = \sum_{t \in T} \frac{1}{3} p_t c^*(t) + \sum_{e \in A} \left( 1 - \sum_{t \in T: e \in t} \frac{1}{3} p_t \right) c_e^*$
Main Lemma

If $\exists \alpha \geq 1$ s.t. $w(t) \leq \alpha c^*(t) \forall t \in T$, then $E[\text{cost}(\pi)] \leq \alpha \text{cost}(\pi^*)$.

Proof: We charge cost of $\pi$-backward edges in $G_w$ to triangles in $T$.

• Expected cost charged to $t \in T$ is

$\left(\frac{1}{3} p_tw_{e_1} + \frac{1}{3} p_tw_{e_2} + \frac{1}{3} p_tw_{e_3}\right) = \frac{1}{3} p_tw(t)$.

$\Rightarrow$

$E[\text{cost}(\pi)] = \sum_{t \in T} \frac{1}{3} p_tw(t) + \sum_{e \in A_w} \left(1 - \sum_{t \in T; e \in t} \frac{1}{3} p_t\right) \bar{w}_e$

$\text{cost}(\pi^*) = \sum_{e \in A_w} c^*_e = \sum_{t \in T} \frac{1}{3} ptc^*(t) + \sum_{e \in A_w} \left(1 - \sum_{t \in T; e \in t} \frac{1}{3} p_t\right) c^*_e$
Upper Bounding $\alpha$

- Probability constraints:
  $\Rightarrow w(t) \leq 5c^*(t)$.

- Triangle constraints:
  $\Rightarrow w(t) \leq 3c^*(t)$.

- Both:
  $\Rightarrow w(t) \leq 2c^*(t)$.

$\Rightarrow$ 2-approximation for Rank Aggregation...
Getting 11/7-Approximation

- Another 2-approximation algorithm for Rank Aggregation:

\[
\text{\textsc{Pick-A-Perm}}(\pi_1, \ldots, \pi_k): \\
\text{return a randomly selected input ranking.}
\]

- Yet another 2-approximation algorithm...?
Getting 11/7-Approximation

- **KwikSort** good when **Pick-A-Perm** bad.
- **Pick-A-Perm** good when **KwikSort** bad.
- **Lemma:**

\[
\frac{3}{7}E[\text{KwikSort}] + \frac{4}{7}E[\text{Pick-A-Perm}] \leq \frac{11}{7}OPT.
\]

⇒ Minimum of **Pick-A-Perm** and **KwikSort** is an 11/7 - approximation for Rank Aggregation.
LP-Based Approach for Rank Aggregation

\[
\text{minimize } \sum_{u \neq v} x_{uv} w_{vu} \text{ s.t. } \\
x_{uv} \leq x_{uy} + x_{yv} \quad \forall u, v, y \in V \\
x_{uv} + x_{vu} = 1 \quad \forall u, v \in V \\
x_{uv} \geq 0 \quad \forall u, v \in V
\]

Solve to get solution \( \{x_{uv}\}_{u,v \in V} \), and then...
Rounding the LP solution

\text{LP-KwikSort}(V, w, x)

pick \( v \in V \) uniformly at random

\( V_L, V_R \leftarrow \emptyset \)

for all \( u \neq v \)

with probability \( x_{uv} \)

add \( u \) to \( V_L \)

else (with remaining prob. \( x_{vu} \))

add \( u \) to \( V_R \)

return order

\( \pi = \text{[LP-KwikSort}(V_L, w, x), v, \text{LP-KwikSort}(V_R, w, x)] \)
Analysis of LP-KwikSort

- Two ways edge is charged:
  - “Safe” way:
    - Expected contribution $x_{uv}w_{vu} + x_{vu}w_{uv} = $ LP contribution.
  - “Dangerous” way:
    - Charged to triplet $t = \{u, v, y\}$. Define $A_t, p_t$...

Expected contribution to $t$ due to $\{u, v\}$:

$$p_t \left[ \frac{1}{3} (x_{uv}x_{vu}w_{vu} + x_{vu}x_{yu}w_{uv}) \right] = p_t \left[ \frac{1}{3} (q^t_{uv}w_{vu} + q^t_{vu}w_{uv}) \right].$$
Analysis of LP-KwikSort

\[ E[LP-KwikSort] = \]
\[ \sum_{t \in T} p_t \frac{1}{3} \sum_{\{u,v\} \subseteq t} (q^t_{uv} w_{vu} + q^t_{vu} w_{uv}) \]
\[ + \sum_{\{u,v\} \subseteq V} \left(1 - \sum_{t: \{u,v\} \subseteq t} p_t \frac{1}{3} (q^t_{uv} + q^t_{vu})\right) (x_{uv} w_{vu} + x_{vu} w_{uv}) \]

LP value = \[\sum_{\{u,v\} \subseteq V} (x_{uv} w_{vu} + x_{vu} w_{uv}) = \]
\[ \sum_{t \in T} p_t \frac{1}{3} \sum_{\{u,v\} \subseteq t} (q^t_{uv} + q^t_{vu}) (x_{uv} w_{vu} + x_{vu} w_{uv}) \]
\[ + \sum_{\{u,v\} \subseteq V} \left(1 - \sum_{t: \{u,v\} \subseteq t} p_t \frac{1}{3} (q^t_{uv} + q^t_{vu})\right) (x_{uv} w_{vu} + x_{vu} w_{uv}) \]
Analysis of LP-KwikSort

• Probability constraints:
  \[ E[LP-KwikSort] \leq 2.5(\text{LP value}). \]

• Triangle inequality constraints:
  \[ ??? \]

• Both:
  \[ E[LP-KwikSort] \leq 2(\text{LP value}). \]
Getting 4/3-Approximation

\[
\frac{2}{3} E[\text{LP-KwikSort}] + \frac{1}{3} E[\text{Pick-A-Perm}] \leq \frac{4}{3} (\text{LP value}).
\]

⇒ Minimum of LP-KwikSort and Pick-A-Perm is a 4/3 - approximation for Rank Aggregation.
KwikSort exp. runtime $T(G) = O(|G| \log |G|)$?

By induction assume $T(G) = 100|G| \log |G|$ for $|G| < n$. Let $|G| = n$. Can assume that in recursion, if left or right parts greater than $99T/100$, recurse first in large sub-recursion. Let $\mathcal{E}$ denote the event that both sides of subrecursions are at most of size $99T/100$.

$$T(G) = E[\#\text{Comparisons for } \mathcal{E}] + \sum 100\alpha_i n \log(\alpha_i n)$$

where $\alpha_1...\alpha_k$ satisfies $\sum \alpha_i = 1$ and $\alpha_i \leq 9/10$ for all $i$.

$$T(G) \leq nE[\#\text{Splits for } \mathcal{E}] + \sum 100n \log n + 100n \log(9/10)$$

Must make sure $E[\#\text{Splits for } \mathcal{E}] = O(1)$. For normal QuickSort this is easy because $E[\#\text{Splits for } \mathcal{E}] = 100/98$ which is killed by $100 \log(99/100)$. But for KwikSort: Let $x$ be size of right recursion. $E[x] = n/2$ (why?) and $Var(x) = n^2/12$ (why?). So by Chebychev $Pr[x > 99n/100] \leq 0.4$ and similarly for left side. So $Pr[\neg \mathcal{E}] \leq 0.8$, $Pr[\mathcal{E}] \geq 0.2$, $E[\#\text{Splits for } \mathcal{E}] \leq 5$. So the answer is YES.