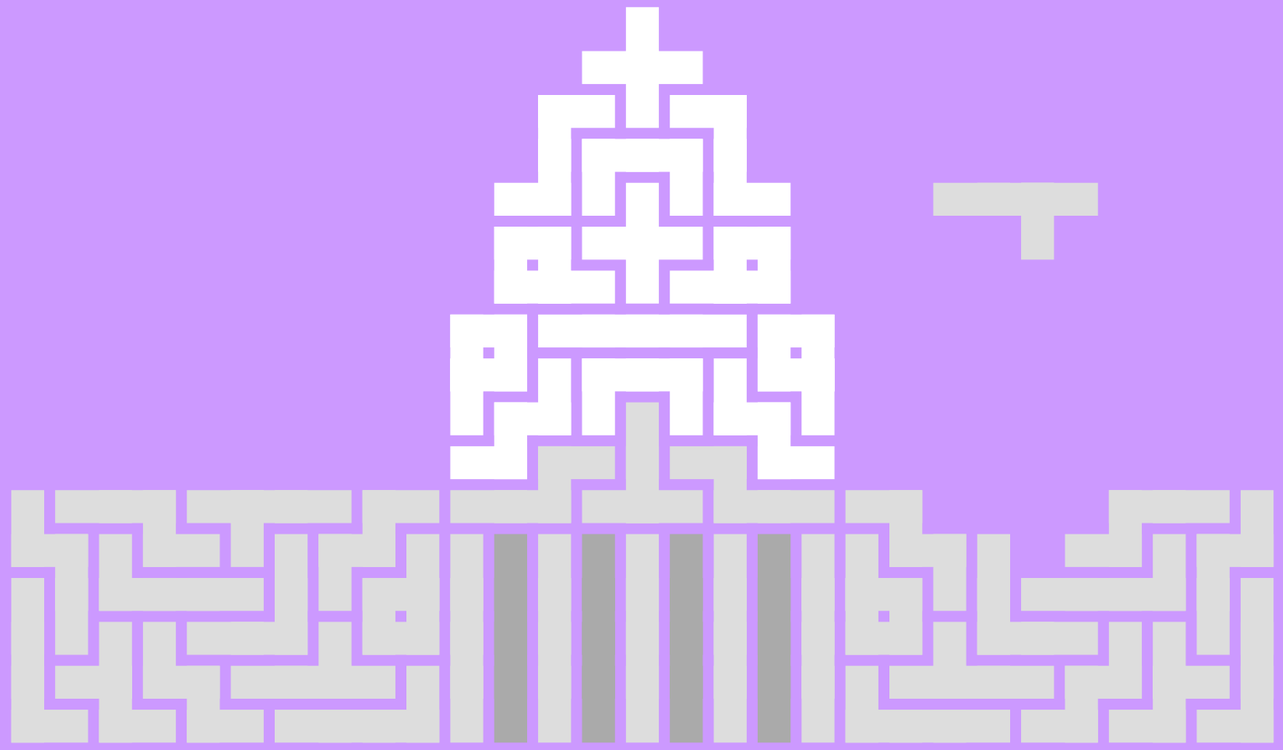


Algorithms For Democratic Decision-Making

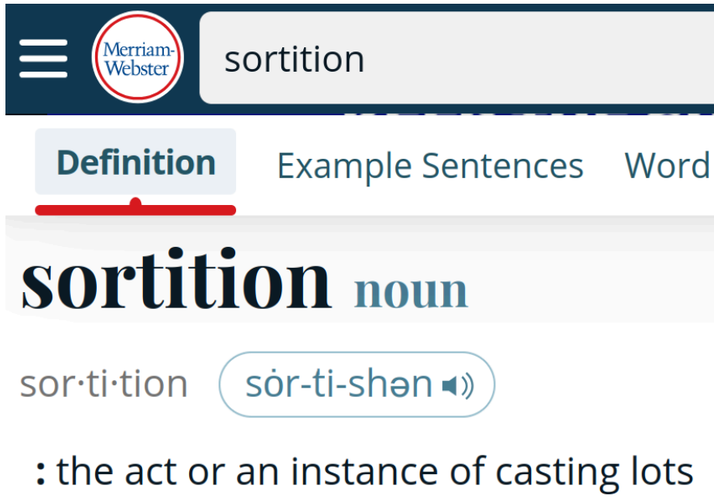
Jamie Tucker-Foltz • Yale University • Spring 2026

Lecture 17: **Citizens Assemblies**

Announcements

Next week I will be at a conference so we will meet over Zoom only at the usual link. Check Canvas for an announcement.

Sortition


Merriam-Webster
sortition
Definition
Example Sentences
Word
sortition noun
sor·ti·tion
sȯr-ti-shən
: the act or an instance of casting lots

Sortition

The New York Times

Democracy Is in Trouble. This Region Is Turning to Its People.

A small corner of Belgium is recruiting ordinary citizens to help create policies. Participants say it's renewed their faith in government.

Listen · 6:30 min Share full article 74

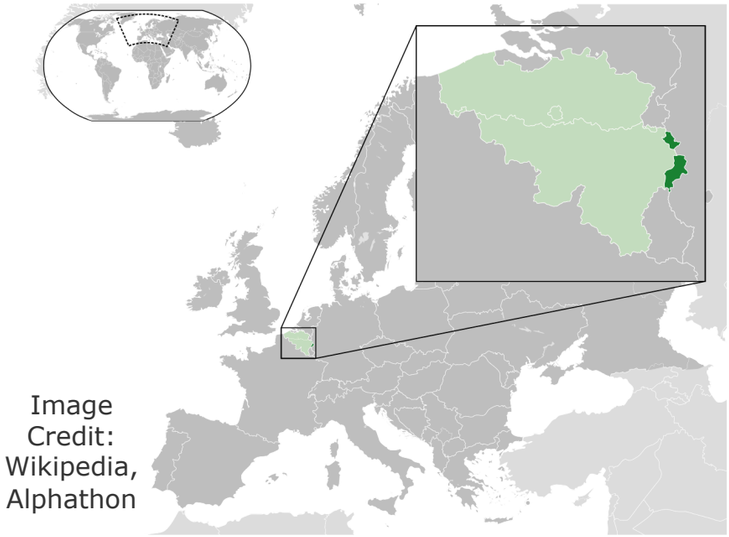


Image Credit: Wikipedia, Alphathon

Merriam-Webster sortition

Definition Example Sentences Word

sortition noun

sor·ti·tion sȯr-ti-shən

: the act or an instance of casting lots

Sortition

The New York Times

Democracy Is in Trouble. This Region Is Turning to Its People.

A small corner of Belgium is recruiting ordinary citizens to help create policies. Participants say it's renewed their faith in government.

Listen · 6:30 min Share full article 74

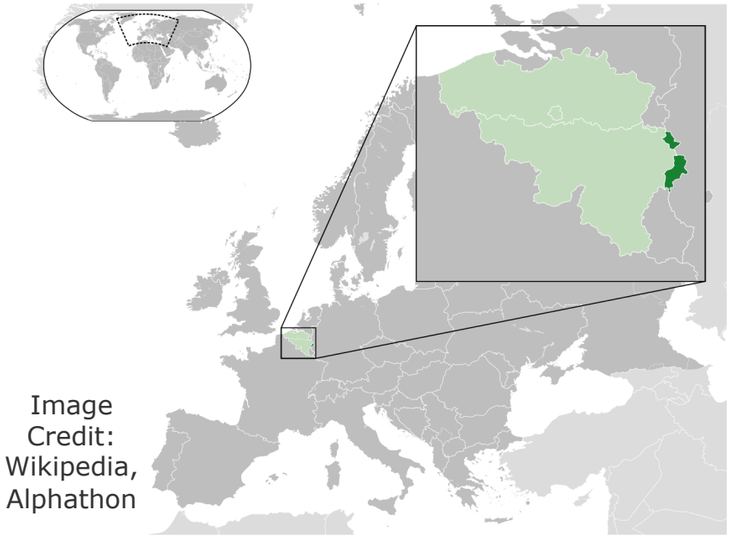


Image Credit: Wikipedia, Alphathon

Merriam-Webster sortition

Definition Example Sentences Word

sortition noun

sor·ti·tion sȯr-ti-shən

: the act or an instance of casting lots

"About 1,500 letters are sent once a year to randomly selected residents in Ostbelgien. Of those who indicate interest, about 30 are chosen to join the citizens' assembly...
Each participant is paid a stipend of about 115 euros ($133) per day...
Though the assemblies' recommendations are not binding, lawmakers are required to consider them, and many have been adopted."

Democracy in Ancient Greece

Democracy in Ancient Greece

Image Credit: Jen Christiansen

Democracy in Ancient Greece

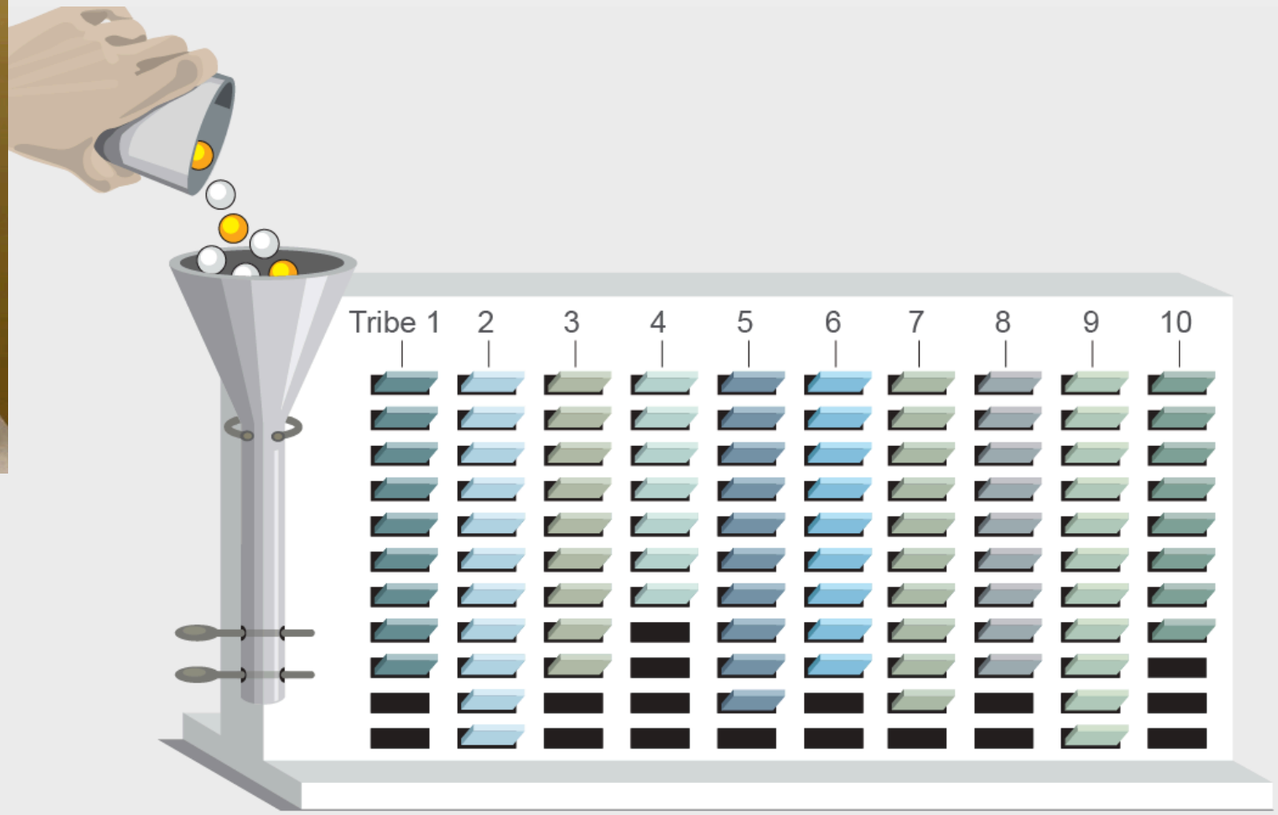


Image Credit: Jen Christiansen

Democracy in Ancient Greece

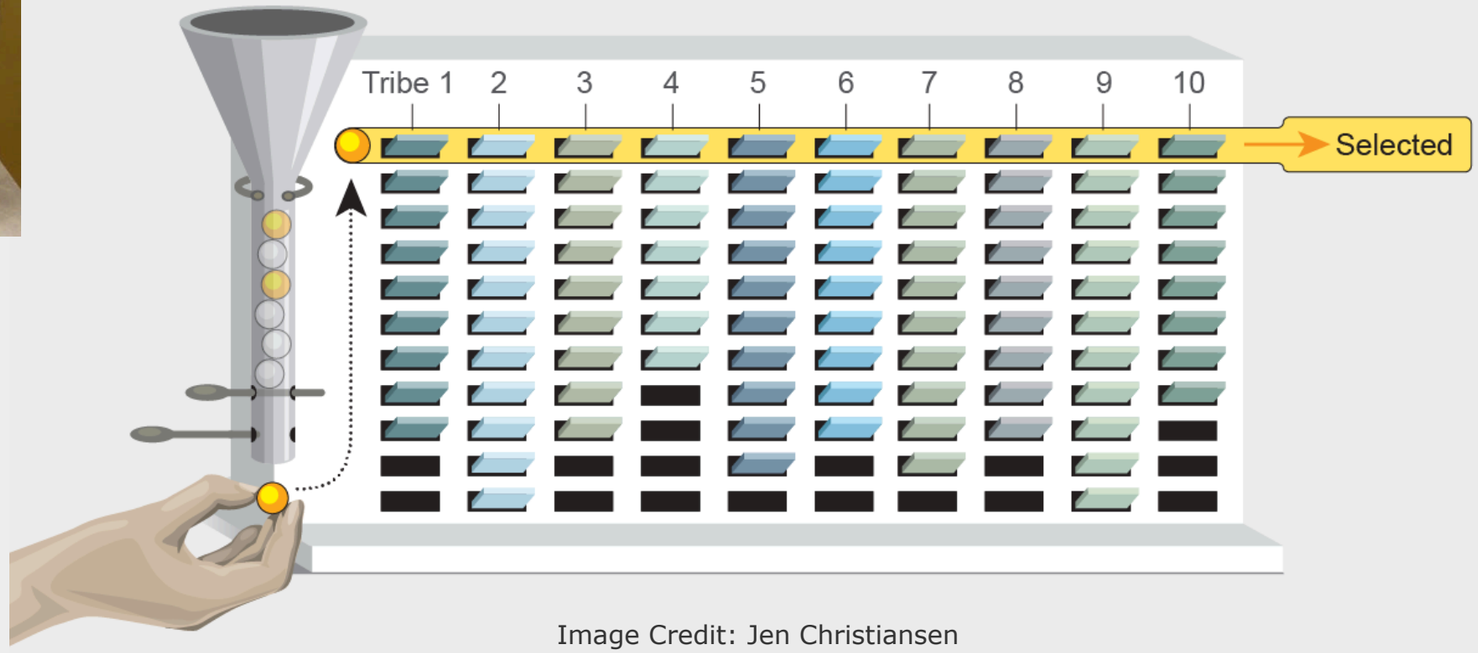


Image Credit: Jen Christiansen

Democracy in Ancient Greece

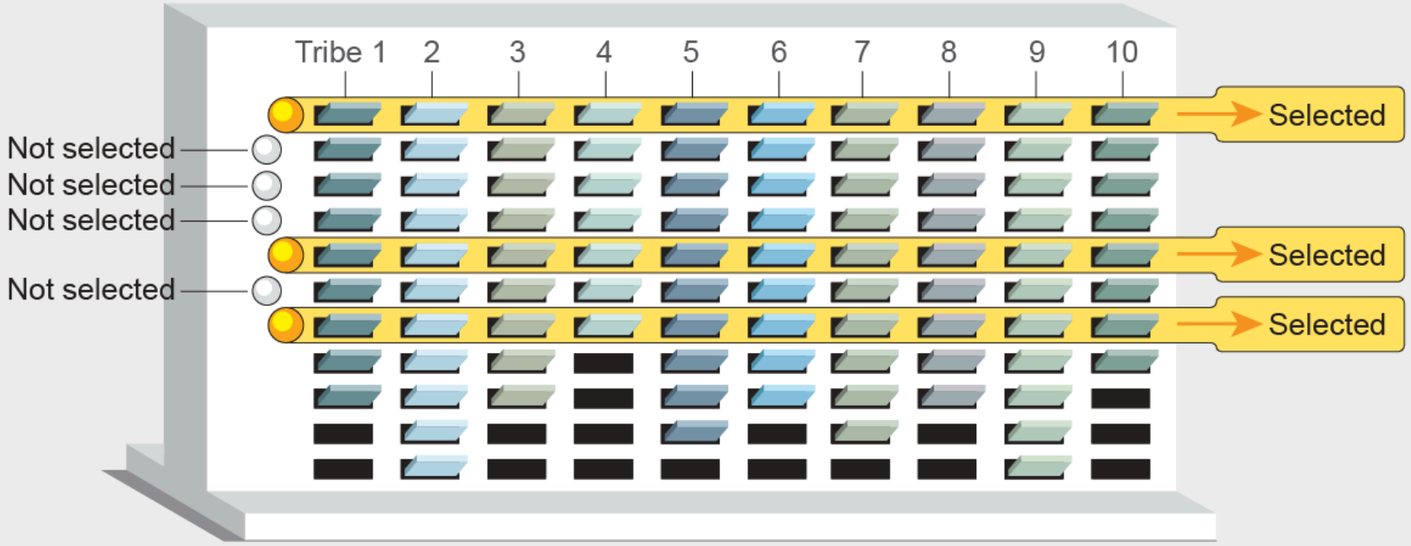


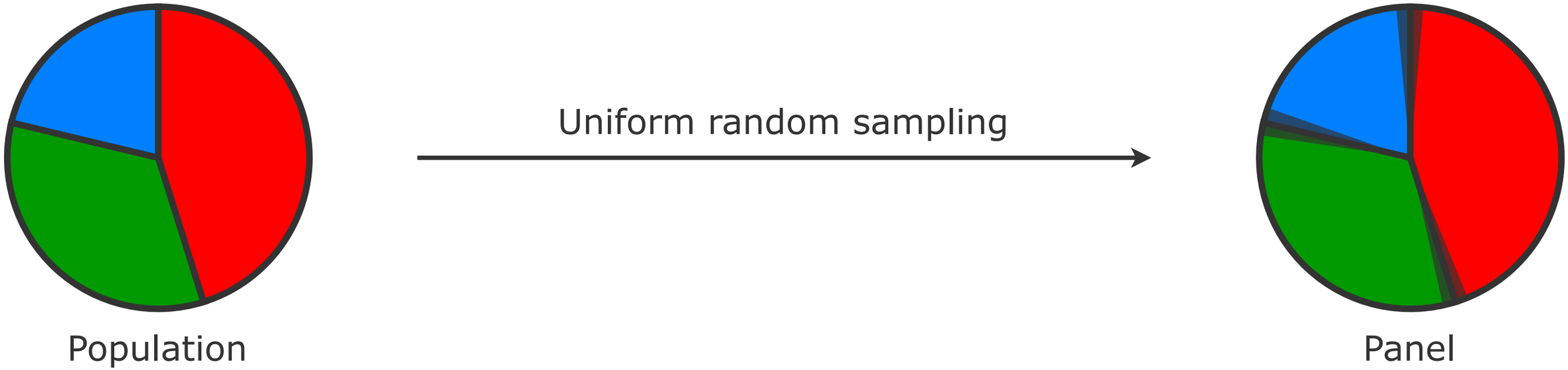
Image Credit: Jen Christiansen

How to select the panel

https://www.youtube.com/watch?v=EDGp5eGnnxI

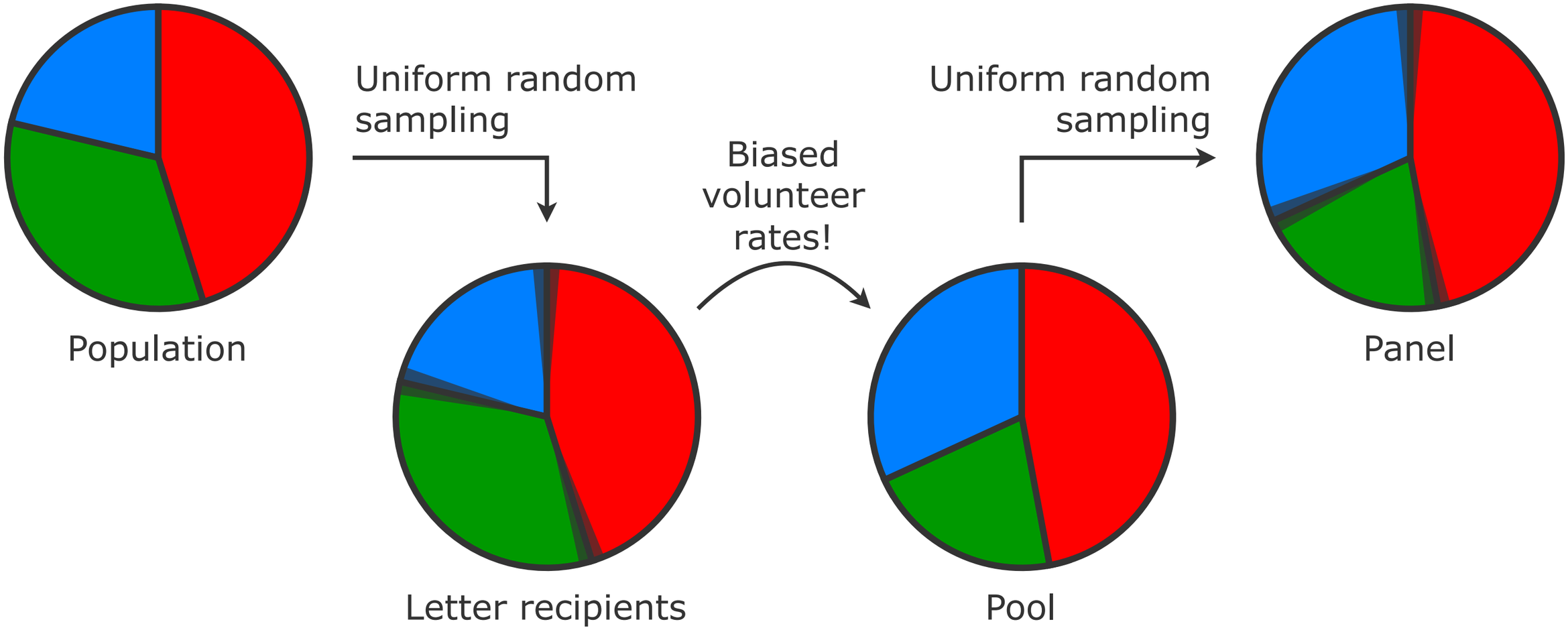
How to select the panel

https://www.youtube.com/watch?v=EDGp5eGnnxI



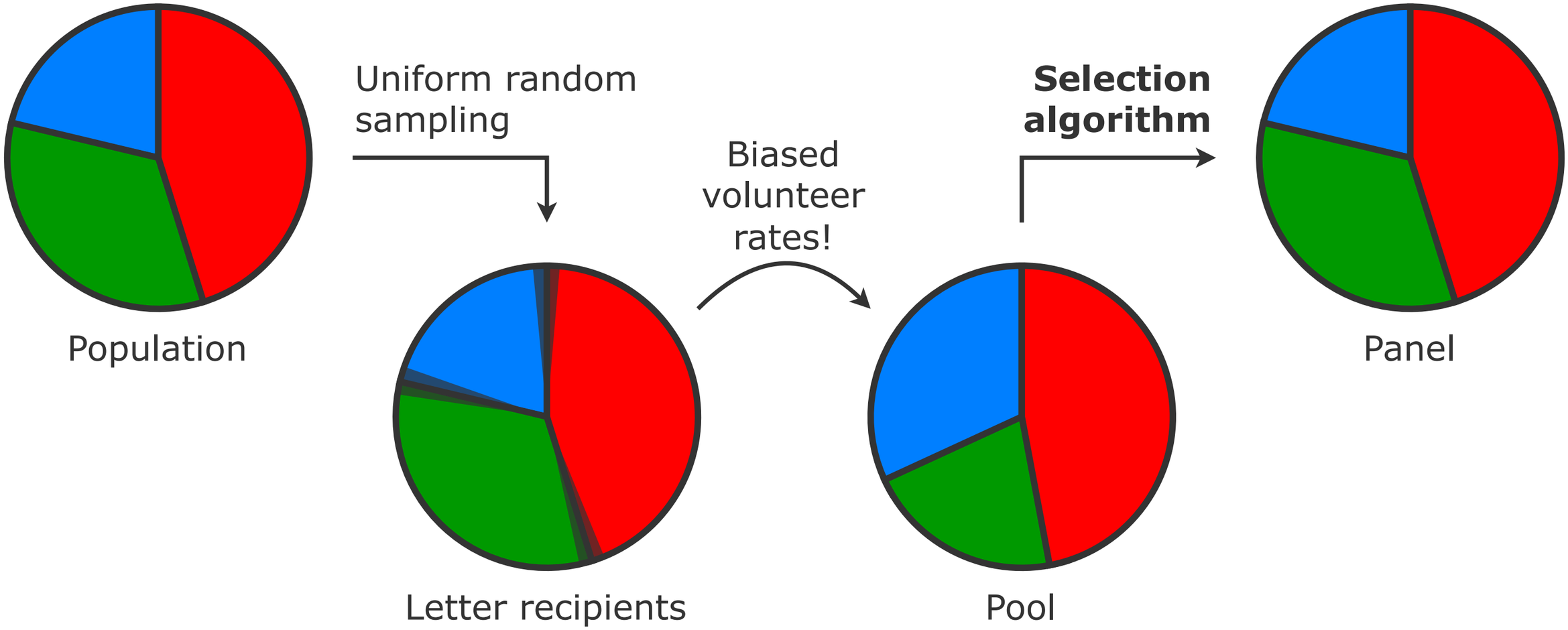
How to select the panel

https://www.youtube.com/watch?v=EDGp5eGnnxI

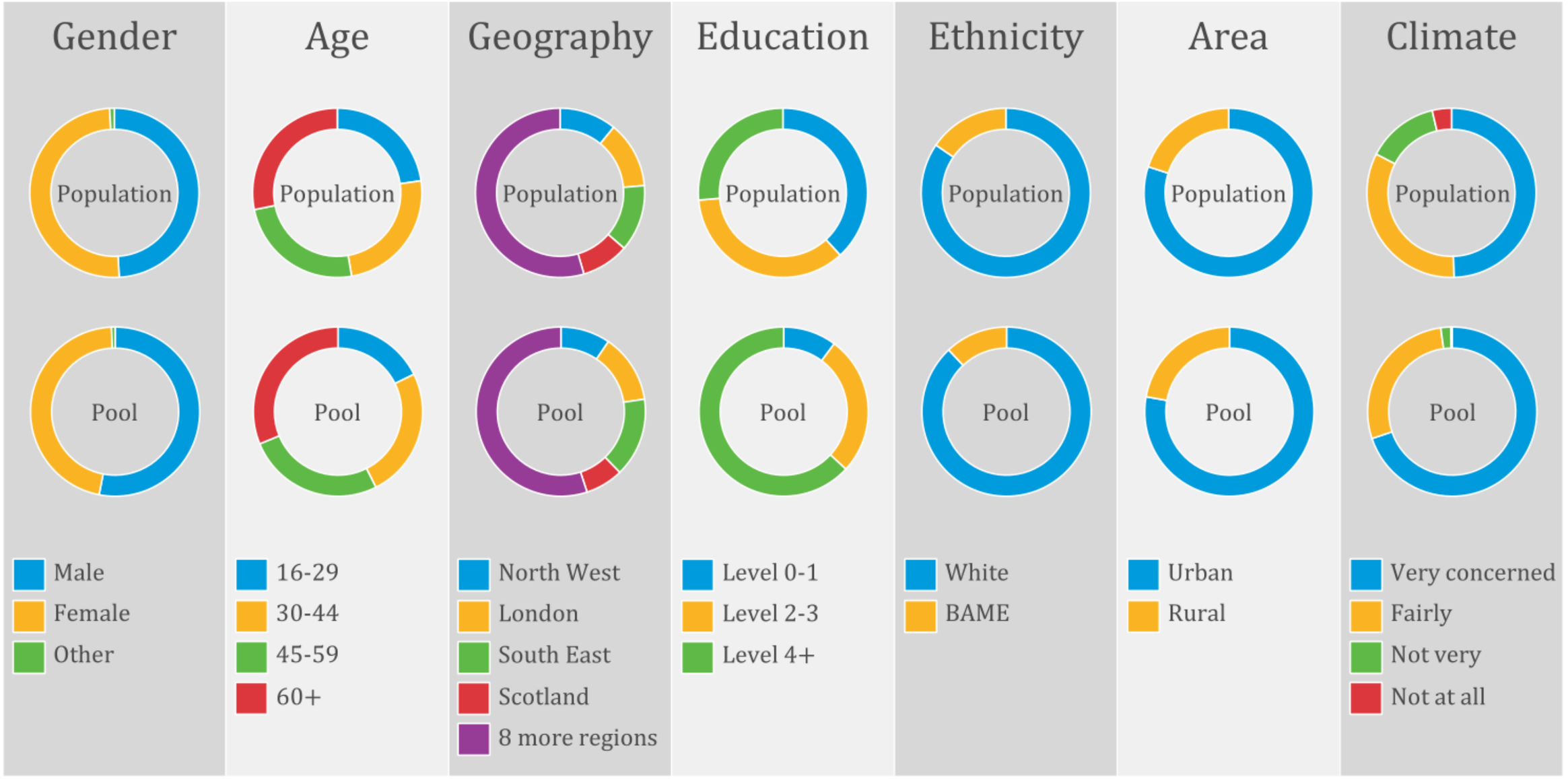


How to select the panel

https://www.youtube.com/watch?v=EDGp5eGnnxI



Data from the UK Climate Assembly



The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$

The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$
- Demographic information: For each *feature* $f \in F$, the set $N_f \subseteq [n]$ has feature f

The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$
- Demographic information: For each *feature* $f \in F$, the set $N_f \subseteq [n]$ has feature f
- Quotas $\ell_f \leq u_f$ for each feature $f \in F$

The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$
- Demographic information: For each *feature* $f \in F$, the set $N_f \subseteq [n]$ has feature f
- Quotas $\ell_f \leq u_f$ for each feature $f \in F$

Output: A distribution over quota-compliant panels, i.e., subsets $K \subseteq [n]$ of size k such that, for each panel K in the distribution and feature f, $\ell_f \leq |K \cap N_f| \leq u_f$.

The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$
- Demographic information: For each *feature* $f \in F$, the set $N_f \subseteq [n]$ has feature f
- Quotas $\ell_f \leq u_f$ for each feature $f \in F$

Output: A distribution over quota-compliant panels, i.e., subsets $K \subseteq [n]$ of size k such that, for each panel K in the distribution and feature f, $\ell_f \leq |K \cap N_f| \leq u_f$.

Proposition

Deciding whether there is any quota-compliant panel is NP-complete.

The computational problem

Input:

- Pool size n; the pool will be denoted by $[n]$
- Panel size $k < n$
- Demographic information: For each *feature* $f \in F$, the set $N_f \subseteq [n]$ has feature f
- Quotas $\ell_f \leq u_f$ for each feature $f \in F$

Output: A distribution over quota-compliant panels, i.e., subsets $K \subseteq [n]$ of size k such that, for each panel K in the distribution and feature f, $\ell_f \leq |K \cap N_f| \leq u_f$.

Proposition

Deciding whether there is any quota-compliant panel is NP-complete.

Note: In practice, this problem is actually easy to solve!

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

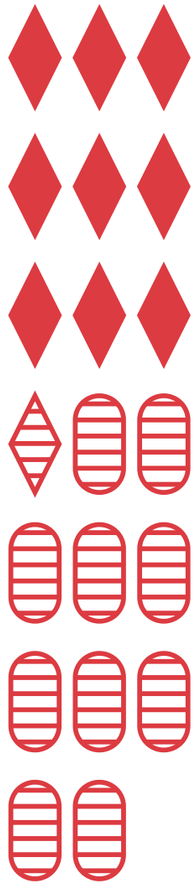
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◇	▯	≡	●
ℓ_f:	4	4	4	4

$n = 20,\ k = 10$

A P



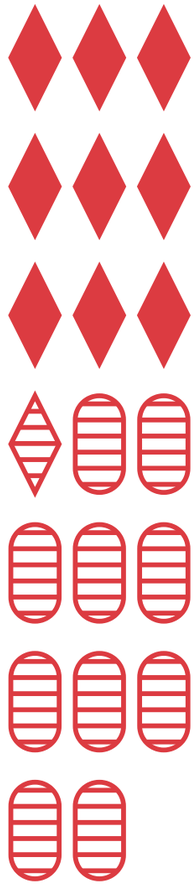
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$			

$n = 20,\ k = 10$

A P



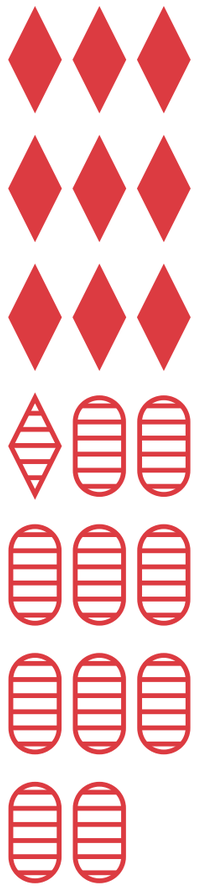
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$	$\frac{4}{10}$		

$n = 20,\ k = 10$

A P



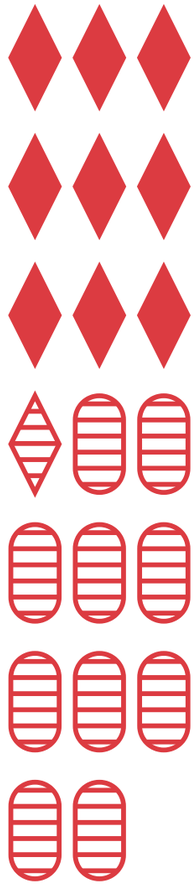
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{4}{11}$	

$n = 20,\ k = 10$

A P



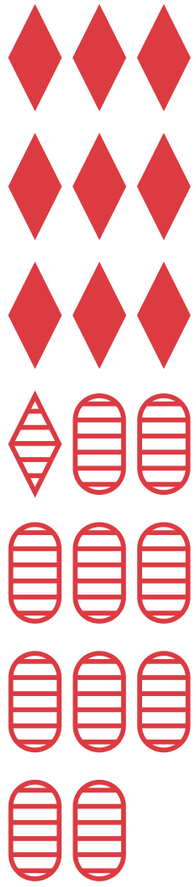
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{4}{9}$

$n = 20,\ k = 10$

A P



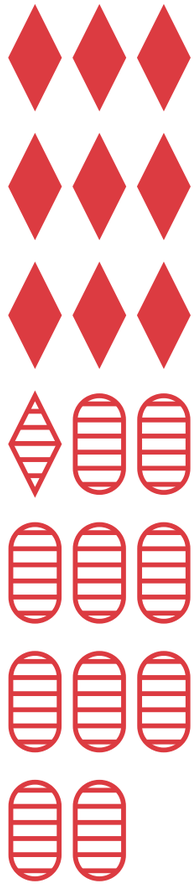
Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{4}{9}$

$n = 20,\ k = 10$

A P



Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{4}{9}$

$n = 20,\ k = 10$

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{3}{9}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{3}{8}$

$n = 20,\ k = 10$

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

$n = 20,\ k = 10$

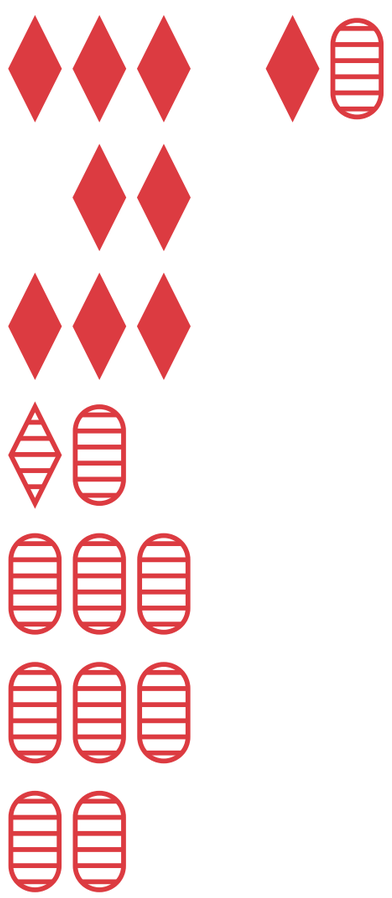
f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{3}{9}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{3}{8}$

Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{3}{9}$	$\frac{4}{10}$	$\frac{4}{11}$	$\frac{3}{8}$

$n = 20,\ k = 10$

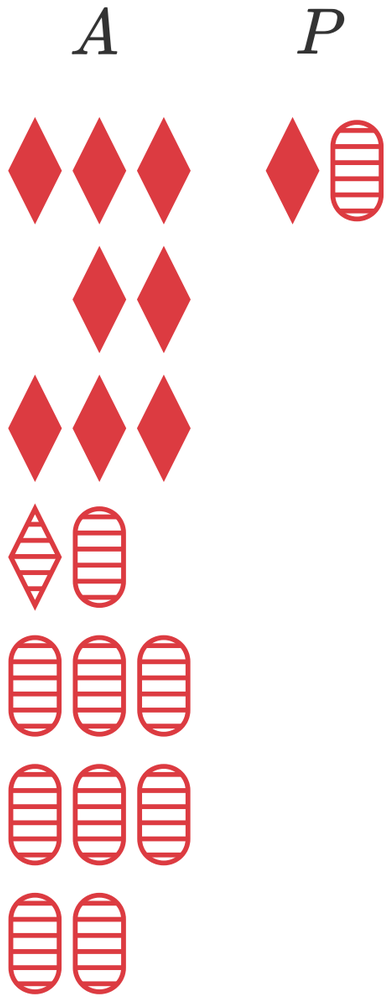


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{10}$	$\frac{3}{8}$

$n = 20,\ k = 10$

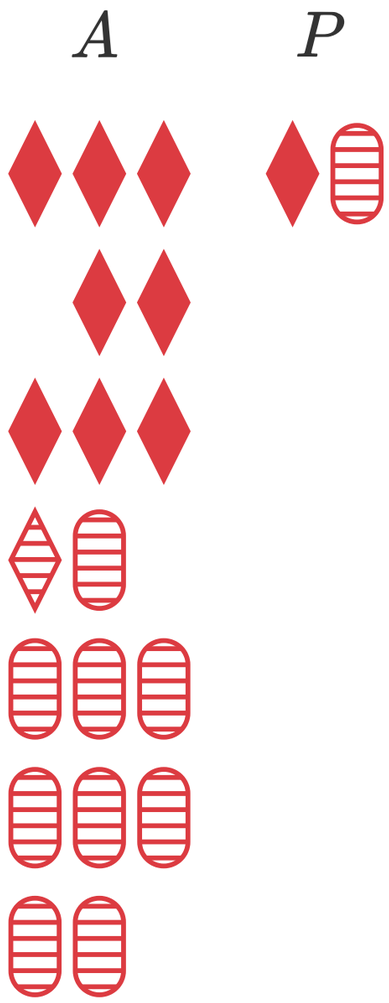


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	$\frac{3}{9}$	$\frac{3}{9}$	$\frac{3}{10}$	$\frac{3}{8}$

$n = 20,\ k = 10$

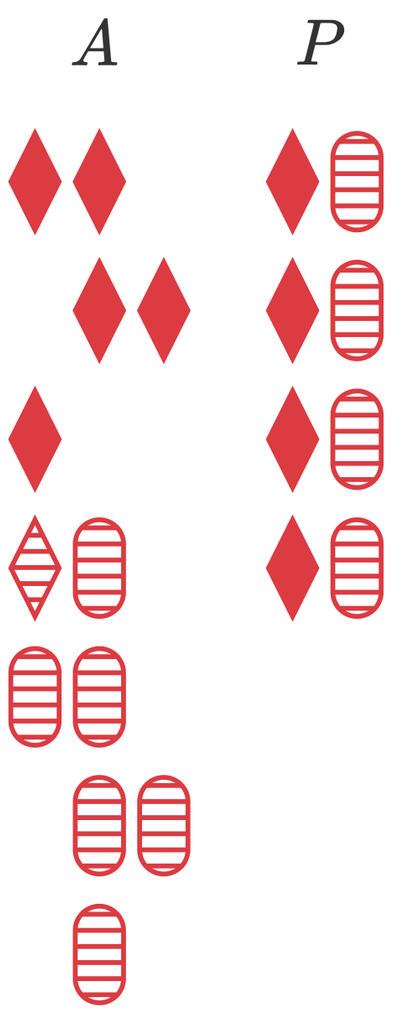


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	–	–	–	–

$n = 20,\ k = 10$

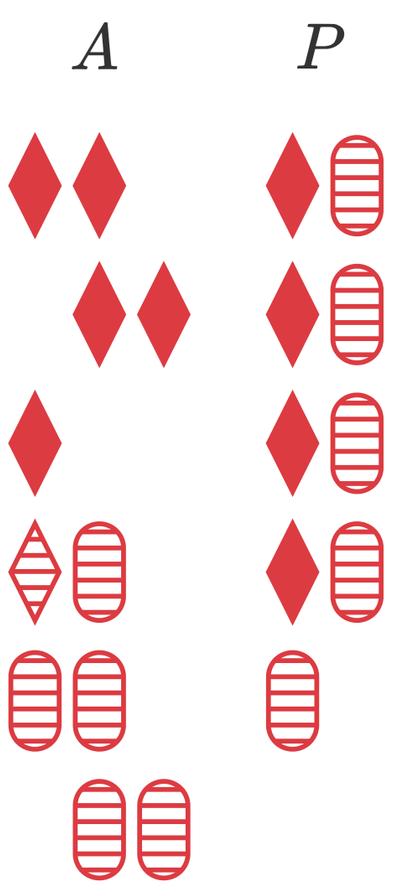


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	—	—	—	—

$n = 20,\ k = 10$

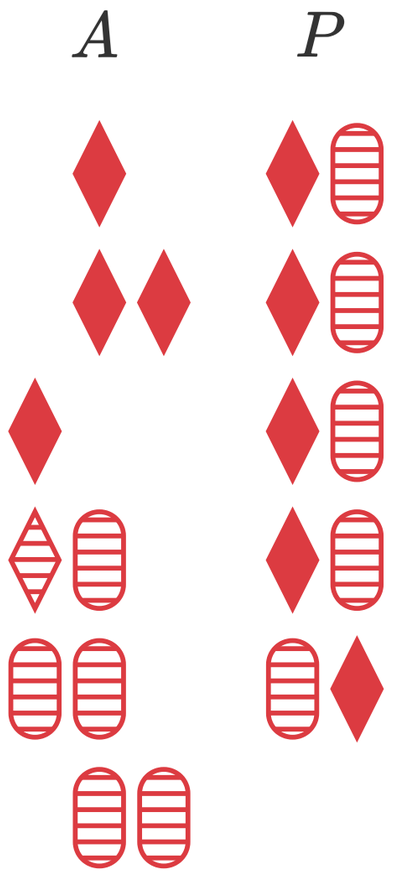


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	—	—	—	—

$n = 20,\ k = 10$

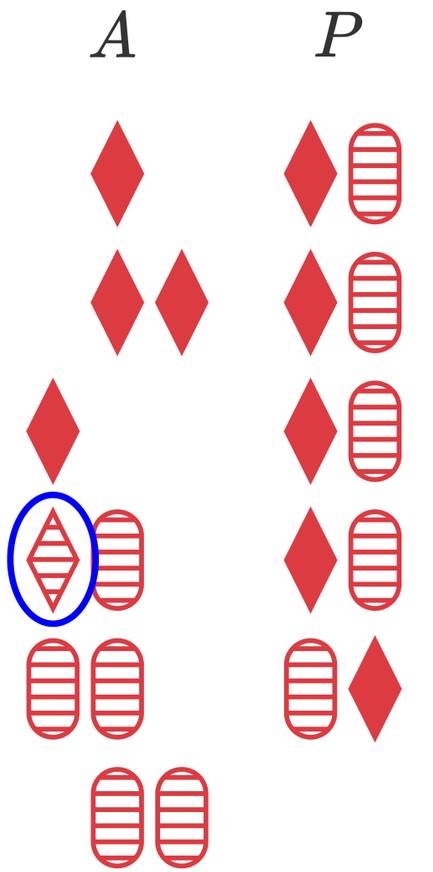


Algorithm 1: Greedy

$P \leftarrow \{\}$
$A \leftarrow [n]$
for $i \in [k]$:
    remove participants from A that would violate quotas if added to P
    if all lower quotas are satisfied (i.e., $|N_f \cap P| \geq \ell_f$ for all $f \in F$):
        $j \leftarrow$ uniformly random participant from A
    else:
        $f \leftarrow$ feature maximizing $\frac{\ell_f - |N_f \cap P|}{|N_f \cap A|}$ over f s.t. $|N_f \cap P| < \ell_f$
        $j \leftarrow$ uniformly random participant from $N_f \cap A$
    if no such j exists:
        start the whole algorithm over
    $A \leftarrow A \setminus \{j\}$
    $P \leftarrow P \cup \{j\}$
return P

f:	◊	▯	≡	●
ℓ_f:	4	4	4	4
Priority:	—	—	—	—

$n = 20,\ k = 10$



Algorithm 2: Leximin

Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected

Algorithm 2: Leximin

Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual

Algorithm 2: Leximin

Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

Algorithm 2: Leximin

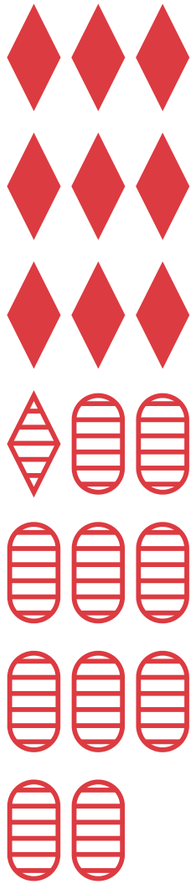
Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

$n = 20,\ k = 10$

f: 

ℓ_f:	4	4	4	4



▸ What is the probability the striped diamond is selected under leximin?

Respond at:

pollev.com/jtuckerfoltz255 or

bit.ly/jtfpoll or

text jtuckerfoltz255 to 37607

Algorithm 2: Leximin

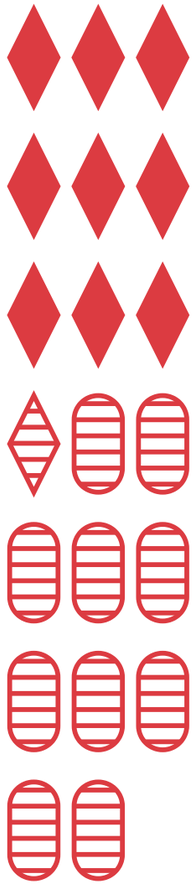
Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

$n = 20,\ k = 10$

f: 

ℓ_f: 4 4 4 4



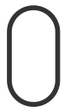
▸ **What is the probability the striped diamond is selected under leximin?** Answer: 1/2.

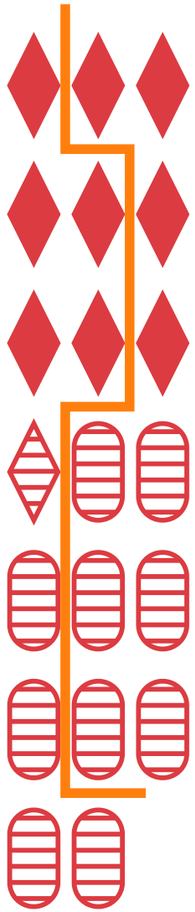
Algorithm 2: Leximin (Flanigan Gölz, Gupta, Hennig, Procaccia, 2021)

Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

$n = 20,\ k = 10$

f:				
ℓ_f:	4	4	4	4



▸ **What is the probability the striped diamond is selected under leximin?** Answer: 1/2.

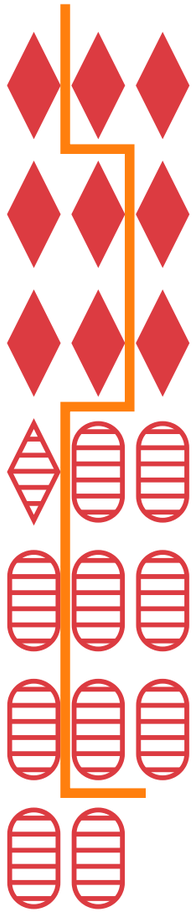
Algorithm 2: Leximin (Flanigan Gölz, Gupta, Hennig, Procaccia, 2021)

Select the optimal lottery over quota-compliant panels in order to maximize:

- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

$n = 20,\ k = 10$

f:	◇	▯	≡	●
ℓ_f:	4	4	4	4



Panelot

▸ **What is the probability the striped diamond is selected under leximin?** Answer: 1/2.

Algorithm 2: Leximin (Flanigan Gölz, Gupta, Hennig, Procaccia, 2021)

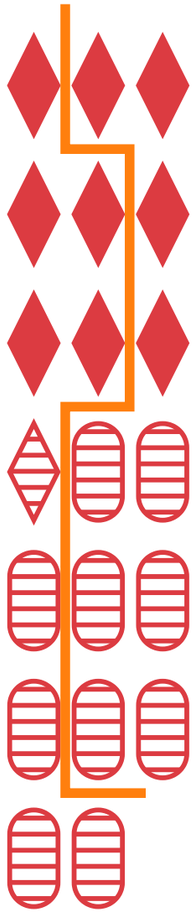
Select the optimal lottery over quota-compliant panels in order to maximize:

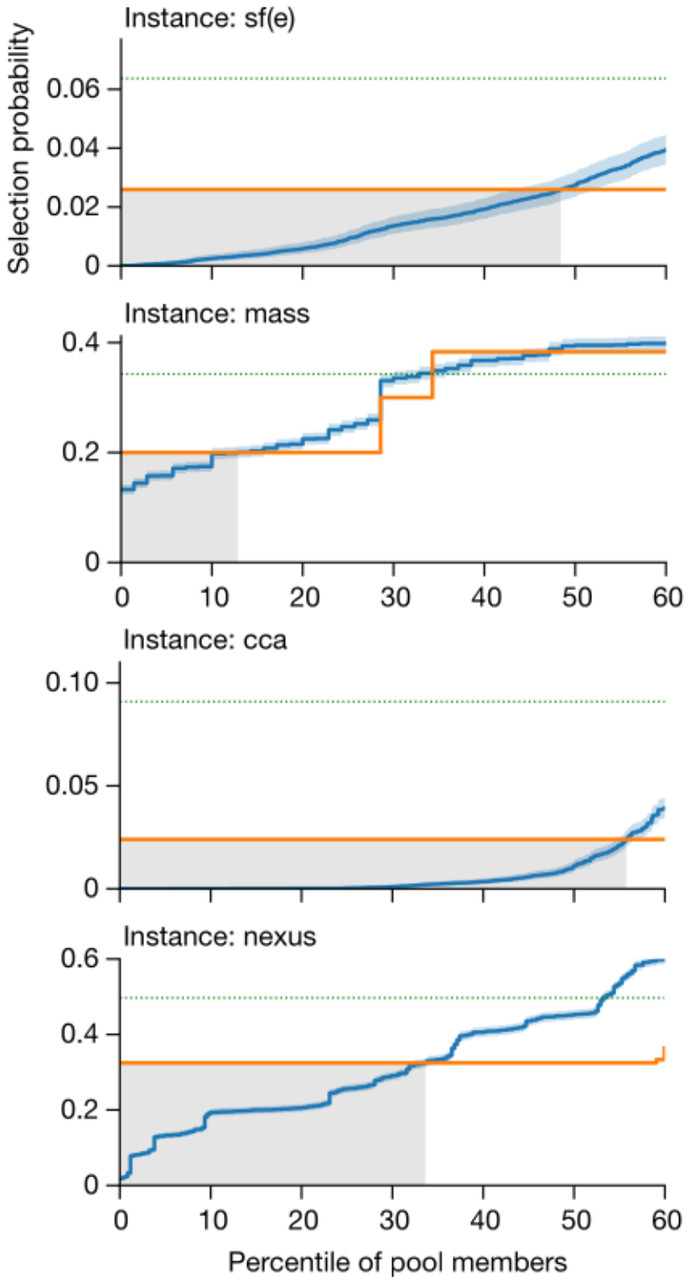
- The minimum probability of any individual from the pool being selected
- Break ties by maximizing the next smallest probability of any individual
- And then the next smallest, and so on

$n = 20,\ k = 10$

f: ◇ ⬭ ≡ ●

ℓ_f: 4 4 4 4





▸ **What is the probability the striped diamond is selected under leximin?** Answer: 1/2.

Strategic behavior

Let $M(A, I)$ be the greatest possible increase in probability of being selected that an individual can gain by misreporting their features for algorithm A on instance I.

Strategic behavior

Let $M(A, I)$ be the greatest possible increase in probability of being selected that an individual can gain by misreporting their features for algorithm A on instance I.

Proposition

Even with only one feature there are instances I for which $M(A, I)$ is arbitrarily close to 1 for any selection algorithm A.

Strategic behavior

Let $M(A, I)$ be the greatest possible increase in probability of being selected that an individual can gain by misreporting their features for algorithm A on instance I.

Proposition

Even with only one feature there are instances I for which $M(A, I)$ is arbitrarily close to 1 for any selection algorithm A.

Proof illustration. Let $n = 100,\ k = 10,$

f: ◊ ⬯

$\ell_f = u_\ell$: 1 9

Strategic behavior

Let $M(A, I)$ be the greatest possible increase in probability of being selected that an individual can gain by misreporting their features for algorithm A on instance I.

Proposition

Even with only one feature there are instances I for which $M(A, I)$ is arbitrarily close to 1 for any selection algorithm A.

Proof illustration. Let $n = 100,\ k = 10,$

f: ◊ ⬭

$\ell_f = u_\ell$: 1 9

A ◊ can pretend to be a ⬭, increasing its probability from 1/90 to 9/11. ■

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

▸ What is the supremum gain from manipulation M(Leximin, I) over all 1/4-rich instances I with two features?

Respond at:

pollev.com/jtuckerfoltz255 or

bit.ly/jtfpoll or

text jtuckerfoltz255 to 37607

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

▸ **What is the supremum gain from manipulation M(Leximin, I) over all 1/4-rich instances I with two features?** Answer: 1

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

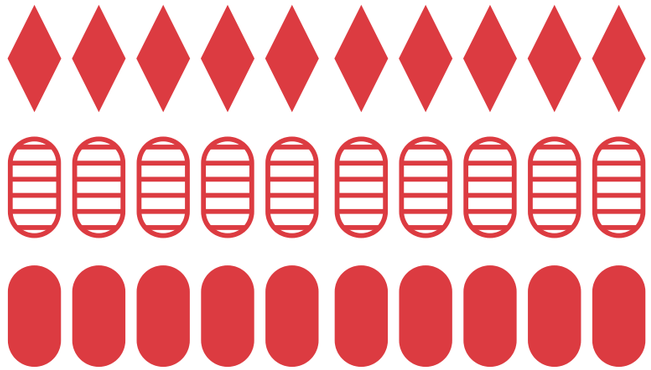
Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

$n = 30,\ k = 4$

$\ell_f = u_f$: 2 2 2 2



▸ **What is the supremum gain from manipulation M(Leximin, I) over all 1/4-rich instances I with two features?** Answer: 1

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

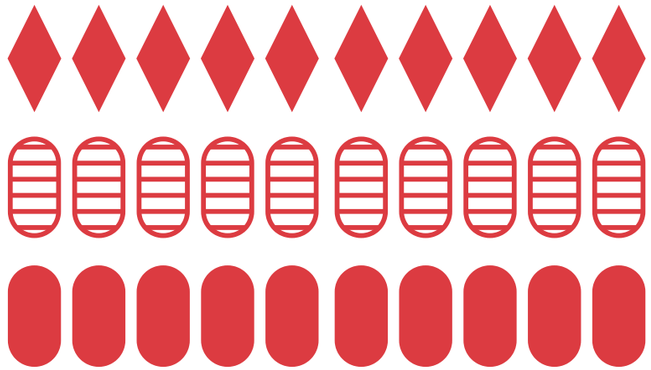
Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

$n = 30,\ k = 4$



What is the supremum gain from manipulation M(Leximin, I) over all 1/4-rich instances I with two features? Answer: 1

Manipulability of Leximin (Flanigan, Liang, Procaccia, Wang, 2024)

Say that an instance is κ-rich if there is a quota-compliant panel supported on participants with feature vectors that each make up at least a κ-fraction of the total population.

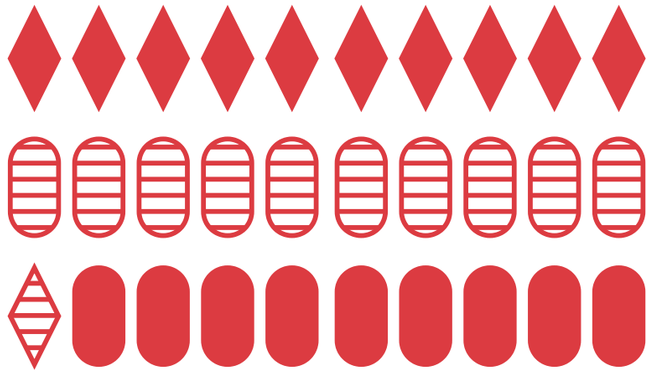
Note: The family of counterexamples from the previous slide is not κ-rich for any constant κ.

Can you construct instances where Leximin is highly manipulable with:

- Two features
- Constant $\kappa = \frac{1}{4}$?

$n = 30,\ k = 4$

$\ell_f = u_f$: 2 2 2 2



Always ~~Never~~ selected!

What is the supremum gain from manipulation M(Leximin, I) over all 1/4-rich instances I with two features? Answer: 1

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Theorem (Flanigan, Liang, Procaccia, Wang, 2024)

For constant κ, over κ-rich instances, Minimax achieves the worst-case asymptotically optimal gain from manipulation of $O(1/n)$.

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Theorem (Flanigan, Liang, Procaccia, Wang, 2024)

For constant κ, over κ-rich instances, Minimax achieves the worst-case asymptotically optimal gain from manipulation of $O(1/n)$.

Proof. Take a quota-compliant panel P and sample uniformly from the set of voters with the same feature vectors as each individual from P.

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Theorem (Flanigan, Liang, Procaccia, Wang, 2024)

For constant κ, over κ-rich instances, Minimax achieves the worst-case asymptotically optimal gain from manipulation of $O(1/n)$.

Proof. Take a quota-compliant panel P and sample uniformly from the set of voters with the same feature vectors as each individual from P.

This gives us a valid lottery over panels where each voter is selected with probability at most $\frac{k}{\kappa n} = O(1/n)$. So Minimax also selects each voter with at most this probability.

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Theorem (Flanigan, Liang, Procaccia, Wang, 2024)

For constant κ, over κ-rich instances, Minimax achieves the worst-case asymptotically optimal gain from manipulation of $O(1/n)$.

Proof. Take a quota-compliant panel P and sample uniformly from the set of voters with the same feature vectors as each individual from P.

This gives us a valid lottery over panels where each voter is selected with probability at most $\frac{k}{\kappa n} = O(1/n)$. So Minimax also selects each voter with at most this probability.

To prove optimality, consider an arbitrary algorithm A on an instance where all participants are [redacted] and all panel members must be [redacted]. Some participant has probability $\geq k/n$.

Algorithm 3: Minimax

Instead of maximizing the minimum probability of selection, minimize the maximum!

Theorem (Flanigan, Liang, Procaccia, Wang, 2024)

For constant κ, over κ-rich instances, Minimax achieves the worst-case asymptotically optimal gain from manipulation of $O(1/n)$.

Proof. Take a quota-compliant panel P and sample uniformly from the set of voters with the same feature vectors as each individual from P.

This gives us a valid lottery over panels where each voter is selected with probability at most $\frac{k}{\kappa n} = O(1/n)$. So Minimax also selects each voter with at most this probability.

To prove optimality, consider an arbitrary algorithm A on an instance where all participants are  and all panel members must be  . Some participant has probability $\geq k/n$.

Then they could have misreported for a gain of k/n if their true type was  . ■

Algorithm 4: Goldilocks (Baharav, Flanigan, 2024)

Minimize $$\frac{(\text{max selection probability})}{k/n} + \gamma \frac{k/n}{(\text{min selection probability})}.$$

Algorithm 4: Goldilocks (Baharav, Flanigan, 2024)

Minimize $$\frac{(\text{max selection probability})}{k/n} + \gamma \frac{k/n}{(\text{min selection probability})}.$$