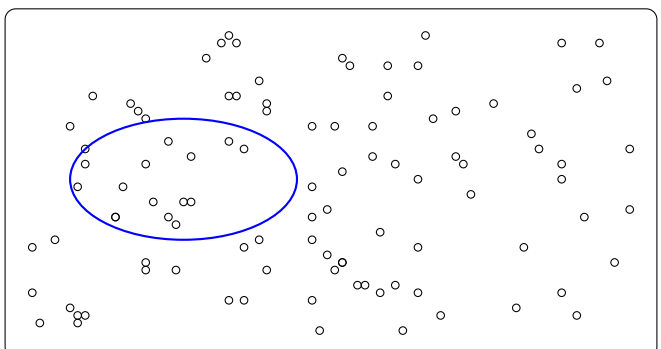
Weight reduction in distributed protocols: new algorithms and analysis

Tolik Zinovyev

Boston University

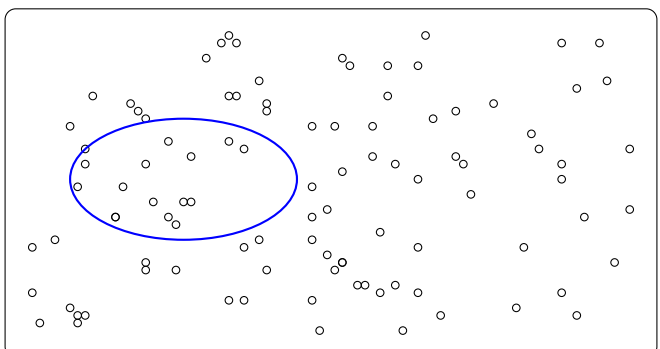
Committee selection

- Have n parties, want to select a small subset and delegate work to them (e.g., in consensus)
- The protocol should remain reliable / secure



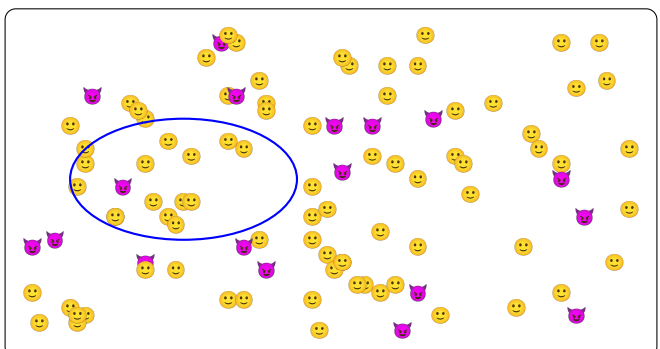
Committee selection

- Have n parties, want to select a small subset and delegate work to them (e.g., in consensus)
- The protocol should remain reliable / secure



Committee selection

- Have n parties, want to select a small subset and delegate work to them (e.g., in consensus)
- The protocol should remain reliable / secure



Committee selection (cont.)

Typical goal:

- 20% of parties are malicious, need to elect a committee with $< \frac{1}{3}$ parties malicious
- some security is lost: $\frac{1}{5} \to \frac{1}{3}$

$$t_i = \begin{cases} 1 \text{ if party } i \text{ is chosen} \\ 0 \text{ otherwise} \end{cases}$$

$A \subseteq [n]$ – adversary set, want

$$\text{for all } A \text{ with } |A| \leq \frac{n}{5} : \quad \Pr\left[\sum_{i \in A} t_i < \frac{1}{3}\sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Typical goal:

- 20% of parties are malicious, need to elect a committee with $< \frac{1}{3}$ parties malicious
- some security is lost: $\frac{1}{5} \to \frac{1}{3}$

$$t_i = \begin{cases} 1 \text{ if party } i \text{ is chosen} \\ 0 \text{ otherwise} \end{cases}$$

$A \subseteq [n]$ – adversary set, want

$$\text{for all } A \text{ with } |A| \leq \frac{n}{5} : \quad \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Typical goal:

- 20% of parties are malicious, need to elect a committee with $< \frac{1}{3}$ parties malicious
- some security is lost: $\frac{1}{5} \to \frac{1}{3}$

$$t_i = \begin{cases} 1 \text{ if party } i \text{ is chosen} \\ 0 \text{ otherwise} \end{cases}$$

$A \subseteq [n]$ – adversary set, want

for all A with $|A| \leq \frac{n}{5}$:

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3}\sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Introducing weights

- the n parties have weights $w_1, w_2, ..., w_n \in \mathbb{Z}_{\geq 0}$
- need to assign new weights $t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$
  - say "party i gets t_i tickets"
- new problem statement: adversary has weight at most 20%; he must have $< \frac{1}{3}$ tickets

need to sample
$t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

for all A with

$$\sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ :$$

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Introducing weights

- the n parties have weights $w_1, w_2, ..., w_n \in \mathbb{Z}_{\geq 0}$
- need to assign new weights $t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$
  - say "party i gets t_i tickets"
- new problem statement: adversary has weight at most 20%; he must have $< \frac{1}{3}$ tickets

need to sample
$t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

for all A with

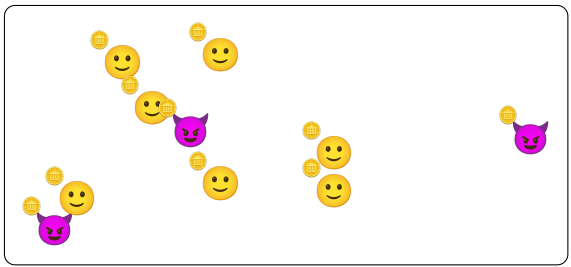
$$\sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ :$$

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Typical solution:

- select each unit of weight with probability p: $t_i \sim \texttt{Binomial}(w_i, p)$



- about 20% of selected weight will be malicious; use tail bounds to analyze bad event
- security error $2^{-\lambda}$ requires committee size $\approx 50\lambda$ (for $\frac{1}{5} \to \frac{1}{3}$)

need to sample
$t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

for all A with

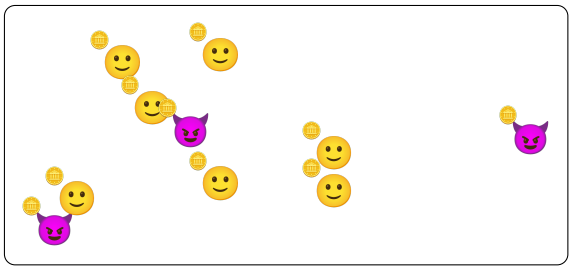
$$\sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ :$$

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Typical solution:

- select each unit of weight with probability p: $t_i \sim \texttt{Binomial}(w_i, p)$



- about 20% of selected weight will be malicious; use tail bounds to analyze bad event
- security error $2^{-\lambda}$ requires committee size $\approx 50\lambda$ (for $\frac{1}{5} \to \frac{1}{3}$)

need to sample $t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

for all A with

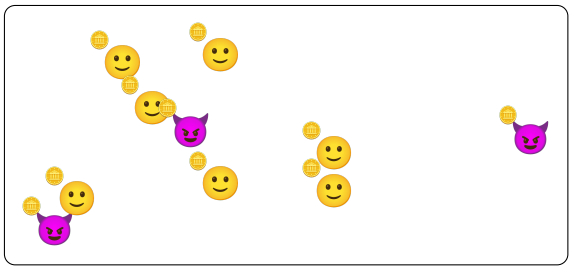
$$\sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ :$$

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Committee selection (cont.)

Typical solution:

- select each unit of weight with probability p: $t_i \sim \texttt{Binomial}(w_i, p)$



- about 20% of selected weight will be malicious; use tail bounds to analyze bad event
- security error $2^{-\lambda}$ requires committee size $\approx 50\lambda$ (for $\frac{1}{5} \to \frac{1}{3}$)

need to sample
$t_1, t_2, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

for all A with

$$\sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ :$$

$$\Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i\right] \text{ is big}$$

Improvements in weighted committee selection

- Take weight distribution $(w_1, ..., w_n)$ into account
- Fait Accompli [GKR23]:
  - selects biggest parties deterministically and others randomly
  - improves committee size vs. error tradeoff
- Swiper [TF23] + others [BHS23, FMT24, DPTX25]:
  - fully deterministic $\Rightarrow$

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ : \ \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i \right] \text{ is big}$$

  - cons: committee size $\Omega(n)$ in worst case (e.g., $w_1 = w_2 = ... = w_n = 1$)
  - pros: security against adaptive corruptions guaranteed
  - pros: (?) deterministic committees are smaller when security parameter is large
- both show that realistic weight distributions allow small committees

Improvements in weighted committee selection

- Take weight distribution $(w_1, ..., w_n)$ into account
- Fait Accompli [GKR23]:
  - selects biggest parties deterministically and others randomly
  - improves committee size vs. error tradeoff
- Swiper [TF23] + others [BHS23, FMT24, DPTX25]:
  - fully deterministic $\Rightarrow$

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ : \ \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i \right] \text{ is big}$$

  - cons: committee size $\Omega(n)$ in worst case (e.g., $w_1 = w_2 = ... = w_n = 1$)
  - pros: security against adaptive corruptions guaranteed
  - pros: (?) deterministic committees are smaller when security parameter is large
- both show that realistic weight distributions allow small committees

Improvements in weighted committee selection

- Take weight distribution $(w_1, ..., w_n)$ into account
- Fait Accompli [GKR23]:
  - selects biggest parties deterministically and others randomly
  - improves committee size vs. error tradeoff
- Swiper [TF23] + others [BHS23, FMT24, DPTX25]:
  - fully deterministic $\Rightarrow$

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ : \ \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i \right] \text{ is big}$$

  - cons: committee size $\Omega(n)$ in worst case (e.g., $w_1 = w_2 = ... = w_n = 1$)
  - pros: security against adaptive corruptions guaranteed
  - pros: (?) deterministic committees are smaller when security parameter is large
- both show that realistic weight distributions allow small committees

Improvements in weighted committee selection

- Take weight distribution $(w_1, ..., w_n)$ into account
- Fait Accompli [GKR23]:
  - selects biggest parties deterministically and others randomly
  - improves committee size vs. error tradeoff
- Swiper [TF23] + others [BHS23, FMT24, DPTX25]:
  - fully deterministic $\Rightarrow$

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ : \ \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i \right] \text{ is big}$$

  - cons: committee size $\Omega(n)$ in worst case (e.g., $w_1 = w_2 = ... = w_n = 1$)
  - pros: security against adaptive corruptions guaranteed
  - pros: (?) deterministic committees are smaller when security parameter is large
- both show that realistic weight distributions allow small committees

Improvements in weighted committee selection

- Take weight distribution $(w_1, ..., w_n)$ into account
- Fait Accompli [GKR23]:
  - selects biggest parties deterministically and others randomly
  - improves committee size vs. error tradeoff
- Swiper [TF23] + others [BHS23, FMT24, DPTX25]:
  - fully deterministic $\Rightarrow$

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \frac{1}{5} \sum_{i=1}^{n} w_i \ : \ \Pr\left[\sum_{i \in A} t_i < \frac{1}{3} \sum_{i=1}^{n} t_i \right] \text{ is big}$$

  - cons: committee size $\Omega(n)$ in worst case (e.g., $w_1 = w_2 = ... = w_n = 1$)
  - pros: security against adaptive corruptions guaranteed
  - pros: (?) deterministic committees are smaller when security parameter is large
- both show that realistic weight distributions allow small committees

Objective function

Minimize the size of the committee $(t_1, t_2, ..., t_n)$: how to quantify?

- the total new weight: $\sum_{i=1}^{n} t_i$
  - useful when the protocol scales poorly with the weights (e.g., secret sharing)

Problem statement

Assume $0 < \alpha < \beta < 1$. Given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

- minimize $\sum_{i=1}^{n} t_i$
- $\alpha \to \beta$, deterministic:

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i \ : \sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$$

This work extends Swiper (Tonkikh, Freitas '24).
Adopt terminology: party i gets t_i "tickets".

- pure optimization problem
- NP-hard? – unknown

Problem statement

Assume $0 < \alpha < \beta < 1$. Given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

- minimize $\sum_{i=1}^{n} t_i$
- $\alpha \to \beta$, deterministic:

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i \ : \sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$$

This work extends Swiper (Tonkikh, Freitas '24).
Adopt terminology: party i gets t_i "tickets".

- pure optimization problem
- NP-hard? – unknown

Problem statement

Assume $0 < \alpha < \beta < 1$. Given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

- minimize $\sum_{i=1}^{n} t_i$
- $\alpha \to \beta$, deterministic:

$$\text{for all } A \text{ with } \sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i \ : \sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$$

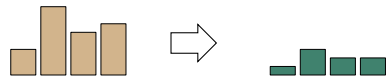
This work extends Swiper (Tonkikh, Freitas '24).
Adopt terminology: party i gets t_i "tickets".

- pure optimization problem
- NP-hard? – unknown

Swiper [TF23] overview

Linear scaling:

- assignment $(t_1, ..., t_n) = (w_1, ..., w_n)$ works, $(t_1, ..., t_n) = 0^n$ doesn't
- set in-between: $t_i = \lfloor s w_i + c \rfloor$ $(c = \text{const})$
- run binary search to find locally minimal s that generates a valid ticket assignment
  - each iteration tests one ticket assignment via dynamic programming for knapsack



Equivalently:

- define potential solutions $\vec{t_1}, \vec{t_2}, ...$ with $\text{size}(\vec{t_j}) = j$ tickets
- find locally minimal j such that $\vec{t_j}$ is valid but $\overrightarrow{t_{j-1}}$ is not

Swiper paper proves that when $c = \alpha$, all $\vec{t_M}, \overrightarrow{t_{M+1}}, ...$ are valid, where

$$M = \left\lfloor \frac{\alpha(1-\alpha)}{\beta - \alpha} n + 1 \right\rfloor = O(n).$$

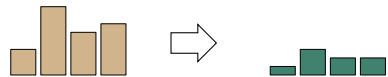
Running time analysis:

- each search iteration: validity testing in time $O\left(n \cdot \sum_{i=1}^{n} t_i\right) \leq O\left(n \cdot M\right) = O\left(n^2\right)$
- total: $O\left(n^2 \log n\right)$

Swiper [TF23] overview

Linear scaling:

- assignment $(t_1, ..., t_n) = (w_1, ..., w_n)$ works, $(t_1, ..., t_n) = 0^n$ doesn't
- set in-between: $t_i = \lfloor s w_i + c \rfloor$ $(c = \text{const})$
- run binary search to find locally minimal s that generates a valid ticket assignment
  - each iteration tests one ticket assignment via dynamic programming for knapsack



Equivalently:

- define potential solutions $\vec{t_1}, \vec{t_2}, ...$ with $\text{size}(\vec{t_j}) = j$ tickets
- find locally minimal j such that $\vec{t_j}$ is valid but $\vec{t_{j-1}}$ is not

Swiper paper proves that when $c = \alpha$, all $\vec{t_M}, \vec{t_{M+1}}, ...$ are valid, where

$$M = \left\lfloor \frac{\alpha(1-\alpha)}{\beta - \alpha} n + 1 \right\rfloor = O(n).$$

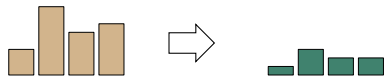
Running time analysis:

- each search iteration: validity testing in time $O\left(n \cdot \sum_{i=1}^{n} t_i\right) \leq O(n \cdot M) = O(n^2)$
- total: $O(n^2 \log n)$

Swiper [TF23] overview

Linear scaling:

- assignment $(t_1, ..., t_n) = (w_1, ..., w_n)$ works, $(t_1, ..., t_n) = 0^n$ doesn't
- set in-between: $t_i = \lfloor s w_i + c \rfloor$ $(c = \text{const})$
- run binary search to find locally minimal s that generates a valid ticket assignment
  - each iteration tests one ticket assignment via dynamic programming for knapsack



Equivalently:

- define potential solutions $\vec{t_1}, \vec{t_2}, ...$ with $\text{size}(\vec{t_j}) = j$ tickets
- find locally minimal j such that $\vec{t_j}$ is valid but $\vec{t_{j-1}}$ is not

Swiper paper proves that when $c = \alpha$, all $\vec{t_M}, \vec{t_{M+1}}, ...$ are valid, where

$$M = \left\lfloor \frac{\alpha(1-\alpha)}{\beta - \alpha} n + 1 \right\rfloor = O(n).$$

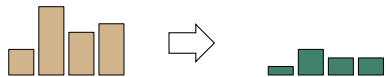
Running time analysis:

- each search iteration: validity testing in time $O\left(n \cdot \sum_{i=1}^{n} t_i\right) \leq O\left(n \cdot M\right) = O\left(n^2\right)$
- total: $O\left(n^2 \log n\right)$

Swiper [TF23] overview

Linear scaling:

- assignment $(t_1, ..., t_n) = (w_1, ..., w_n)$ works, $(t_1, ..., t_n) = 0^n$ doesn't
- set in-between: $t_i = \lfloor s w_i + c \rfloor$ $(c = \text{const})$
- run binary search to find locally minimal s that generates a valid ticket assignment
  - each iteration tests one ticket assignment via dynamic programming for knapsack



Equivalently:

- define potential solutions $\vec{t_1}, \vec{t_2}, ...$ with $\text{size}(\vec{t_j}) = j$ tickets
- find locally minimal j such that $\vec{t_j}$ is valid but $\vec{t_{j-1}}$ is not

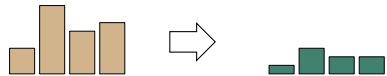
Swiper paper proves that when $c = \alpha$, all $\vec{t_M}, \vec{t_{M+1}}, ...$ are valid, where

$$M = \left\lfloor \frac{\alpha(1-\alpha)}{\beta - \alpha} n + 1 \right\rfloor = O(n).$$

Running time analysis:

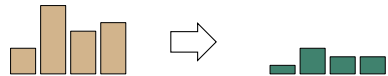
- each search iteration: validity testing in time $O\big(n \cdot \sum_{i=1}^{n} t_i\big) \leq O\big(n \cdot M\big) = O\big(n^2\big)$
- total: $O\big(n^2 \log n\big)$

Improving Swiper: *super Swiper*



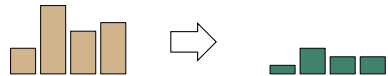
- Replaces binary search with linear search
  - $j \leftarrow 1$;
    while $\overrightarrow{t_j}$ is not valid **do**
    $\quad j \leftarrow j + 1$;
    return $\overrightarrow{t_j}$;
  - binary search can get stuck in a "local minimum"
  - linear search improves output
- reduces running time from $O(n^2 \log n)$ to $O(n + R^2 \log R)$, where $R \leq O(n)$ is the number of tickets $\sum_{i=1}^{n} t_i$ in the output
  - not worse than before
  - normally, $R \ll n \implies n + R^2 \log R \ll n^2 \log n$

Improving Swiper: *super Swiper*



- Replaces binary search with linear search
  - $j \leftarrow 1$;
    while $\overrightarrow{t_j}$ is not valid **do**
    $j \leftarrow j + 1$;
    return $\overrightarrow{t_j}$;
  - binary search can get stuck in a "local minimum"
  - linear search improves output
- reduces running time from $O(n^2 \log n)$ to $O(n + R^2 \log R)$, where $R \leq O(n)$ is the number of tickets $\sum_{i=1}^{n} t_i$ in the output
  - not worse than before
  - normally, $R \ll n \implies n + R^2 \log R \ll n^2 \log n$

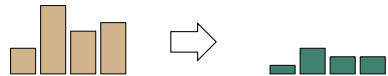
Improving Swiper: *super Swiper*



- Replaces binary search with linear search
  - $j \leftarrow 1$;
    while $\overrightarrow{t_j}$ is not valid **do**
    $\quad j \leftarrow j + 1$;
    return $\overrightarrow{t_j}$;

    Challenge 1: compute $\overrightarrow{t_j}$
    Challenge 2: test $\overrightarrow{t_j}$
  - binary search can get stuck in a "local minimum"
  - linear search improves output
- reduces running time from $O(n^2 \log n)$ to $O(n + R^2 \log R)$, where $R \leq O(n)$ is the number of tickets $\sum_{i=1}^{n} t_i$ in the output
  - not worse than before
  - normally, $R \ll n \implies n + R^2 \log R \ll n^2 \log n$

Improving Swiper: *super Swiper*



- Replaces binary search with linear search
  - $j \leftarrow 1$;
    while $\overrightarrow{t_j}$ is not valid **do**
    $\quad j \leftarrow j + 1$;
    return $\overrightarrow{t_j}$;

    Challenge 1: compute $\overrightarrow{t_j}$ – time $O(n + R \log R)$
    Challenge 2: test $\overrightarrow{t_j}$
  - binary search can get stuck in a "local minimum"
  - linear search improves output
- reduces running time from $O(n^2 \log n)$ to $O(n + R^2 \log R)$, where $R \leq O(n)$ is the number of tickets $\sum_{i=1}^{n} t_i$ in the output
  - not worse than before
  - normally, $R \ll n \implies n + R^2 \log R \ll n^2 \log n$

Testing $\vec{t_j}$

- Reuse dynamic programming computations for testing ticket assignments
- generalized data structure DP:

method	mutable?	parameters	output
.apply(w, t)	mutable	party's weight w_i, # tickets t_i	none
.get()	immutable	none	integer

- guarantee: want to test $\vec{t}$
  - call DP.apply$\big(w_i, (\vec{t})_i\big)$ once for all $i \in [n]$ with $(\vec{t})_i \neq 0$ in any order
  - then DP.get() returns max # tickets the adversary gets in $\vec{t}$
- time complexity
  - if $\sum_{i=1}^{n} (\vec{t})_i = T$, then DP.apply$\big(w_i, (\vec{t})_i\big)$ takes time $O(T)$
  - DP.get() takes time $O(1)$

Testing $\vec{t_j}$

- Reuse dynamic programming computations for testing ticket assignments
- generalized data structure `DP`:

method	mutable?	parameters	output
`.apply`(w, t)	mutable	party's weight w_i, # tickets t_i	none
`.get`$()$	immutable	none	integer

- guarantee: want to test $\vec{t}$
  - call `DP.apply`$\left(w_i, (\vec{t})_i\right)$ once for all $i \in [n]$ with $(\vec{t})_i \neq 0$ in any order
  - then `DP.get`$()$ returns max # tickets the adversary gets in $\vec{t}$
- time complexity
  - if $\sum_{i=1}^{n} (\vec{t})_i = T$, then `DP.apply`$\left(w_i, (\vec{t})_i\right)$ takes time $O(T)$
  - `DP.get`$()$ takes time $O(1)$

Testing $\vec{t_j}$

- Reuse dynamic programming computations for testing ticket assignments
- generalized data structure `DP`:

method	mutable?	parameters	output
`.apply`(w, t)	mutable	party's weight w_i, # tickets t_i	none
`.get`$()$	immutable	none	integer

- guarantee: want to test $\vec{t}$
  - call `DP.apply`$\left(w_i, (\vec{t})_i\right)$ once for all $i \in [n]$ with $(\vec{t})_i \neq 0$ in any order
  - then `DP.get`$()$ returns max # tickets the adversary gets in $\vec{t}$
- time complexity
  - if $\sum_{i=1}^{n} (\vec{t})_i = T$, then `DP.apply`$\left(w_i, (\vec{t})_i\right)$ takes time $O(T)$
  - `DP.get`$()$ takes time $O(1)$

Testing $\vec{t_j}$

- Reuse dynamic programming computations for testing ticket assignments
- generalized data structure `DP`:

method	mutable?	parameters	output
`.apply`(w, t)	mutable	party's weight w_i, # tickets t_i	none
`.get`$()$	immutable	none	integer

- guarantee: want to test $\vec{t}$
  - call `DP.apply`$\left(w_i, (\vec{t})_i\right)$ once for all $i \in [n]$ with $(\vec{t})_i \neq 0$ in any order
  - then `DP.get`$()$ returns max # tickets the adversary gets in $\vec{t}$
- time complexity
  - if $\sum_{i=1}^{n} (\vec{t})_i = T$, then `DP.apply`$\left(w_i, (\vec{t})_i\right)$ takes time $O(T)$
  - `DP.get`$()$ takes time $O(1)$

Testing $\vec{t_j}$ (cont.)

Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$

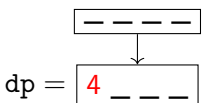
Testing $\vec{t_j}$ (cont.)

Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$

dp = $\boxed{- - - - -}$

Testing $\vec{t_j}$ (cont.)

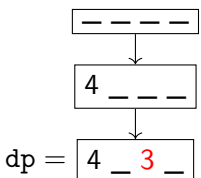
Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$



`dp.apply`$(90, 4)$

Testing $\vec{t_j}$ (cont.)

Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$

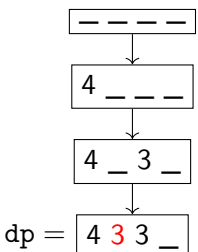


`dp.apply`$(90, 4)$

`dp.apply`$(50, 3)$

Testing $\vec{t_j}$ (cont.)

Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$



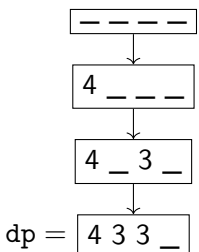
`dp.apply`$(90, 4)$

`dp.apply`$(50, 3)$

`dp.apply`$(60, 3)$

Testing $\vec{t_j}$ (cont.)

Example: want to test $\vec{t} = (4, 3, 3, 0)$ for $\vec{w} = (90, 60, 50, 10)$



`dp.apply`$(90, 4)$

`dp.apply`$(50, 3)$

`dp.apply`$(60, 3)$

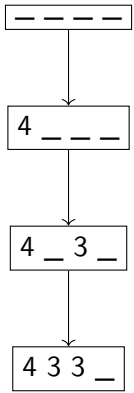
`dp.get`$()$

Testing $\vec{t_j}$ (cont.)

More interesting example: want to test $\vec{t} = (4, 3, 3, 0)$ and $\vec{t'} = (4, 4, 3, 0)$

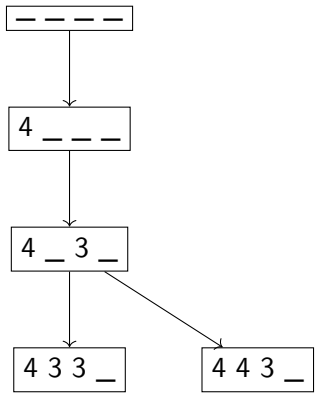
Testing $\vec{t_j}$ (cont.)

More interesting example: want to test $\vec{t} = (4, 3, 3, 0)$ and $\vec{t'} = (4, 4, 3, 0)$



Testing $\vec{t_j}$ (cont.)

More interesting example: want to test $\vec{t} = (4, 3, 3, 0)$ and $\vec{t'} = (4, 4, 3, 0)$



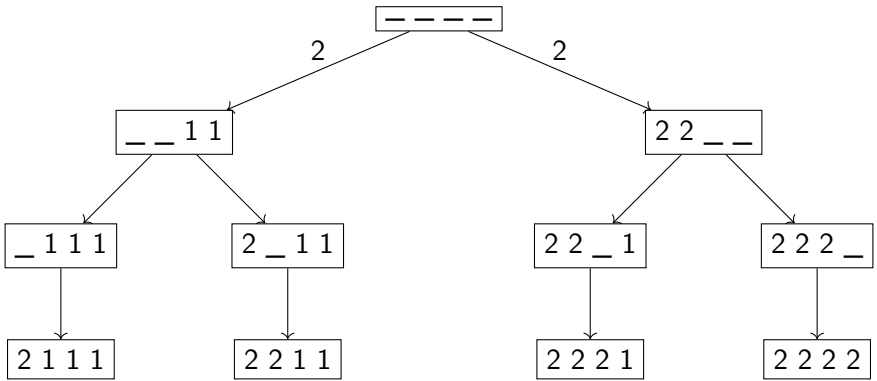
Testing $\vec{t_j}$ (cont.)

More interesting example: want to test $(2,1,1,1)$, $(2,2,1,1)$, $(2,2,2,1)$, $(2,2,2,2)$

$4 \cdot (\log 4 + 1) = 12$ applies. Generalize: can test Swiper's $\vec{t_1}, \vec{t_2}, ..., \overrightarrow{t_{O(R)}}$ with $O(R \log R)$ applies; each apply takes time $O(R)$.

Testing $\vec{t_j}$ (cont.)

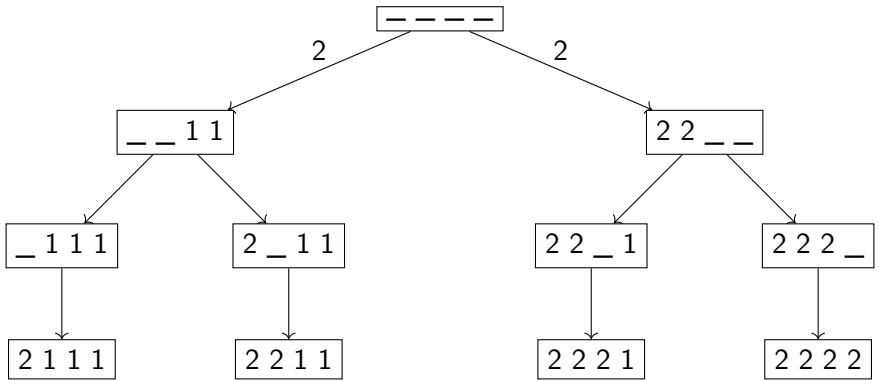
More interesting example: want to test $(2,1,1,1)$, $(2,2,1,1)$, $(2,2,2,1)$, $(2,2,2,2)$



$4 \cdot (\log 4 + 1) = 12$ applies. Generalize: can test Swiper's $\vec{t_1}, \vec{t_2}, ..., \overrightarrow{t_{O(R)}}$ with $O(R \log R)$ applies; each apply takes time $O(R)$.

Testing $\vec{t_j}$ (cont.)

More interesting example: want to test $(2,1,1,1)$, $(2,2,1,1)$, $(2,2,2,1)$, $(2,2,2,2)$



$4 \cdot (\log 4 + 1) = 12$ applies. Generalize: can test Swiper's $\vec{t_1}, \vec{t_2}, ..., \overrightarrow{t_{O(R)}}$ with $O(R \log R)$ applies; each apply takes time $O(R)$.

Improving Swiper: *super Swiper* (cont.)

$j \leftarrow 1$;
while $\overrightarrow{t_j}$ is not valid **do**
  $j \leftarrow j + 1$;
return $\overrightarrow{t_j}$;

- Challenge 1: compute $\overrightarrow{t_j}$ – time $O(n + R \log R)$
- Challenge 2: test $\overrightarrow{t_j}$ – time $O(R^2 \log R)$

Total: time $O(n + R^2 \log R)$

Improving Swiper: *super Swiper* (cont.)

- linear search outputs fewer tickets
- linear search faster than binary search

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	22	85	346	23	29	95	279
	1.61ms	2.07ms	1.96ms	1.40ms	1.88ms	1.80ms	1.96ms
super Swiper	22	58	277	23	29	95	241
	6.55µs	13.0µs	116µs	6.43µs	7.13µs	31.3µs	88.4µs

Table: Evaluation on the Aptos stake distribution

Improving Swiper: *super Swiper* (cont.)

- linear search outputs fewer tickets
- linear search faster than binary search

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	22	85	346	23	29	95	279
	1.61ms	2.07ms	1.96ms	1.40ms	1.88ms	1.80ms	1.96ms
super Swiper	22	58	277	23	29	95	241
	6.55µs	13.0µs	116µs	6.43µs	7.13µs	31.3µs	88.4µs

Table: Evaluation on the Aptos stake distribution

Improving Swiper: *super Swiper* (cont.)

- linear search outputs fewer tickets
- linear search faster than binary search

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	235	745	22393	203	383	1203	17591
	1.10s	1.22s	1.78s	1.03s	1.14s	1.19s	1.65s
super Swiper	232	745	22384	201	383	1171	17581
	75.7µs	461µs	115ms	84.5µs	202µs	1.41ms	118ms

Table: Evaluation on the Algorand stake distribution

Lower bounds

- *super Swiper* works great in practice, what about in theory?
  - how big is output compared to optimal solution?
- our result: *super Swiper*, Swiper [TF23] + others [BHS23, FMT24, DPTX25] all have approximation factor $\Omega(n)$
  - constructed an example where output is $\Omega(n)$ but $\mathrm{OPT} = O(1)$

Lower bounds

- *super Swiper* works great in practice, what about in theory?
  - how big is output compared to optimal solution?
- our result: *super Swiper*, Swiper [TF23] + others [BHS23, FMT24, DPTX25] all have approximation factor $\Omega(n)$
  - constructed an example where output is $\Omega(n)$ but $\mathrm{OPT} = O(1)$

Algorithms with better guarantees

Some progress (see paper):

- exact algorithm (approximation factor 1) with time $O\left(n + R^{3/2} e^{C\sqrt{R}}\right)$
  - $R \leq O(n)$ – size of optimal solution
  - $C = \pi\sqrt{6}/3 \approx 2.57$
  - practical when $R < 100$
- polytime algorithm with approximation factor $O\left(n/\log^2 n\right)$
- polytime algorithm based on linear programming
  - denote the smallest number of tickets by $\mathrm{OPT}_{\alpha,\beta}$
  - LP algorithm outputs solution of size $\mathrm{OPT}_{\alpha,(1-\delta)\beta}$ for any constant $\delta > 0$ ("bi-criteria approximation")

Algorithms with better guarantees

Some progress (see paper):

- exact algorithm (approximation factor 1) with time $O\left(n + R^{3/2} e^{C\sqrt{R}}\right)$
  - $R \leq O(n)$ – size of optimal solution
  - $C = \pi\sqrt{6}/3 \approx 2.57$
  - practical when $R < 100$
- polytime algorithm with approximation factor $O\left(n/\log^2 n\right)$
- polytime algorithm based on linear programming
  - denote the smallest number of tickets by $\mathrm{OPT}_{\alpha,\beta}$
  - LP algorithm outputs solution of size $\mathrm{OPT}_{\alpha,(1-\delta)\beta}$ for any constant $\delta > 0$ ("bi-criteria approximation")

Algorithms with better guarantees

Some progress (see paper):

- exact algorithm (approximation factor 1) with time $O\left(n + R^{3/2} e^{C\sqrt{R}}\right)$
  - $R \leq O(n)$ – size of optimal solution
  - $C = \pi\sqrt{6}/3 \approx 2.57$
  - practical when $R < 100$
- polytime algorithm with approximation factor $O\left(n/\log^2 n\right)$
- polytime algorithm based on linear programming
  - denote the smallest number of tickets by $\mathrm{OPT}_{\alpha,\beta}$
  - LP algorithm outputs solution of size $\mathrm{OPT}_{\alpha,(1-\delta)\beta}$ for any constant $\delta > 0$ ("bi-criteria approximation")

Numerical evaluation

- sub-exponential time exact algorithm can be practical!
- *super Swiper* is almost optimal in practice

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	22	85	346	23	29	95	279
	1.61ms	2.07ms	1.96ms	1.40ms	1.88ms	1.80ms	1.96ms
super Swiper	22	58	277	23	29	95	241
	6.55µs	13.0µs	116µs	6.43µs	7.13µs	31.3µs	88.4µs
exact	22	55		23	29	91	
	133µs	127ms		236µs	1.08ms	45.9s	

Table: Evaluation on the Aptos stake distribution

Numerical evaluation

- sub-exponential time exact algorithm can be practical!
- *super Swiper* is almost optimal in practice

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	22	85	346	23	29	95	279
	1.61ms	2.07ms	1.96ms	1.40ms	1.88ms	1.80ms	1.96ms
super Swiper	22	58	277	23	29	95	241
	6.55µs	13.0µs	116µs	6.43µs	7.13µs	31.3µs	88.4µs
exact	22	55		23	29	91	
	133µs	127ms		236µs	1.08ms	45.9s	

Table: Evaluation on the Aptos stake distribution

Numerical evaluation

- sub-exponential time exact algorithm can be practical!
- *super Swiper* is almost optimal in practice

	$\beta = 1/3$			$\beta = 1/2$			
	$\alpha = 0.2$	$\alpha = 0.25$	$\alpha = 0.3$	$\alpha = 0.3$	$\alpha = 0.35$	$\alpha = 0.4$	$\alpha = 0.45$
Swiper	22	85	346	23	29	95	279
	1.61ms	2.07ms	1.96ms	1.40ms	1.88ms	1.80ms	1.96ms
super Swiper	22	58	277	23	29	95	241
	6.55µs	13.0µs	116µs	6.43µs	7.13µs	31.3µs	88.4µs
exact	22	55		23	29	91	
	133µs	127ms		236µs	1.08ms	45.9s	

Table: Evaluation on the Aptos stake distribution

Conclusion

- Good practical improvements
  - *super Swiper*: better solution in less time: $O\big(n + R^2 \log R\big)$
  - exact algorithm in time $O(n) + 2^{O(\sqrt{R})}$ – sometimes practical
- more theoretical work to be done
  - NP-hard?
  - better approximation factor in polytime
- see `eprint.iacr.org/2025/1076` for latest version

Weight reduction problem:
given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that

- minimize $\sum_{i=1}^{n} t_i$
- for all A with $\sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i$: $\sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$

Conclusion

- Good practical improvements
  - *super Swiper*: better solution in less time: $O\big(n + R^2 \log R\big)$
  - exact algorithm in time $O(n) + 2^{O(\sqrt{R})}$ – sometimes practical
- more theoretical work to be done
  - NP-hard?
  - better approximation factor in polytime
- see eprint.iacr.org/2025/1076 for latest version

Weight reduction problem:
given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that
- minimize $\sum_{i=1}^{n} t_i$
- for all A with $\sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i$: $\sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$

Conclusion

- Good practical improvements
  - *super Swiper*: better solution in less time: $O\big(n + R^2 \log R\big)$
  - exact algorithm in time $O(n) + 2^{O(\sqrt{R})}$ – sometimes practical
- more theoretical work to be done
  - NP-hard?
  - better approximation factor in polytime
- see `eprint.iacr.org/2025/1076` for latest version

Weight reduction problem:
given $w_1, ..., w_n \in \mathbb{Z}_{\geq 0}$, find $t_1, ..., t_n \in \mathbb{Z}_{\geq 0}$ such that
- minimize $\sum_{i=1}^{n} t_i$
- for all A with $\sum_{i \in A} w_i \leq \alpha \sum_{i=1}^{n} w_i$: $\sum_{i \in A} t_i < \beta \sum_{i=1}^{n} t_i$

Fabrice Benhamouda, Shai Halevi, and Lev Stambler.
Weighted secret sharing from wiretap channels.
In Kai-Min Chung, editor, *ITC 2023*, volume 267 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl, June 2023.
doi:10.4230/LIPIcs.ITC.2023.8.

Sourav Das, Benny Pinkas, Alin Tomescu, and Zhuolun Xiang.
Distributed randomness using weighted VUFs.
In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VII*, volume 15607 of *LNCS*, pages 314–344. Springer, Cham, May 2025.
doi:10.1007/978-3-031-91098-2_12.

Hanwen Feng, Tiancheng Mai, and Qiang Tang.
Scalable and adaptively secure any-trust distributed key generation and all-hands checkpointing.
In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2636–2650. ACM Press, October 2024.
doi:10.1145/3658644.3690253.

Peter Gazi, Aggelos Kiayias, and Alexander Russell.
Fait accompli committee selection: Improving the size-security tradeoff of stake-based committees.
In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 845–858. ACM Press, November 2023.
doi:10.1145/3576915.3623194.

Andrei Tonkikh and Luciano Freitas.
Swiper: a new paradigm for efficient weighted distributed protocols.
Cryptology ePrint Archive, Report 2023/1164, 2023.
URL: https://eprint.iacr.org/2023/1164.