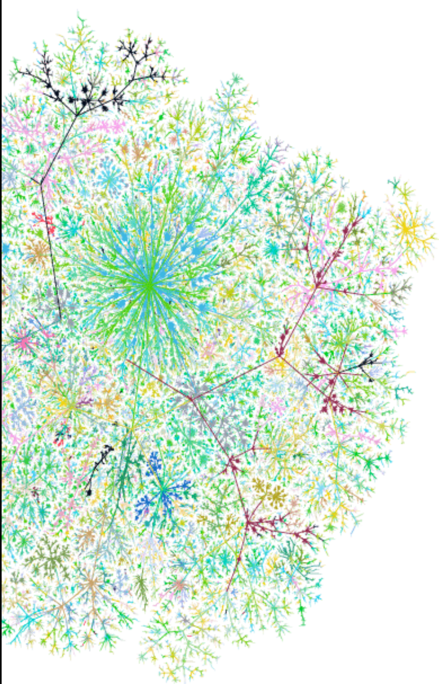
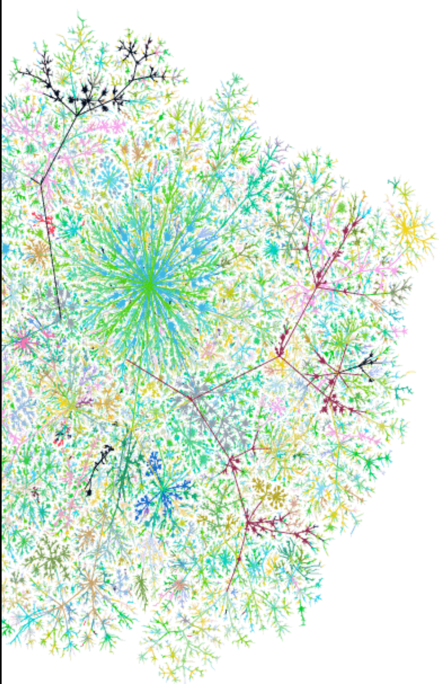


Max Coverage *via* Simplification, Streaming, Sampling, *and* Submodular Stuff

Andrew McGregor
University of Massachusetts

includes joint work with **Hoa Vu**, **David Tench**, **Amit Chakrabarti**, & **Anthony Wirth**

Hi to anyone reading Clement's live tweets!



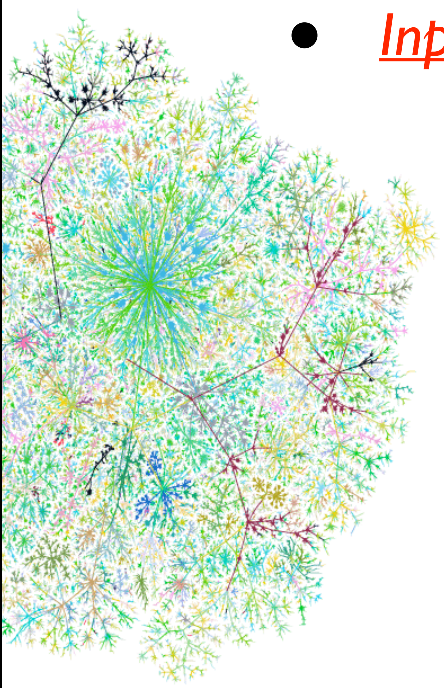
Max Coverage *via* Simplification, Streaming, Sampling, *and* Submodular Stuff

Andrew McGregor

University of Massachusetts

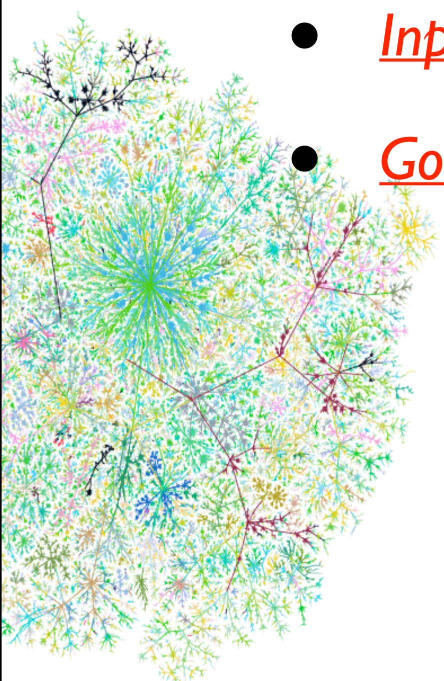
includes joint work with **Hoa Vu**, **David Tench**, **Amit Chakrabarti**, & **Anthony Wirth**

MaxCover Problem



- ***Input*** Sets $S_1, S_2, \ldots, S_m \subseteq [n]$ and integer k

MaxCover Problem

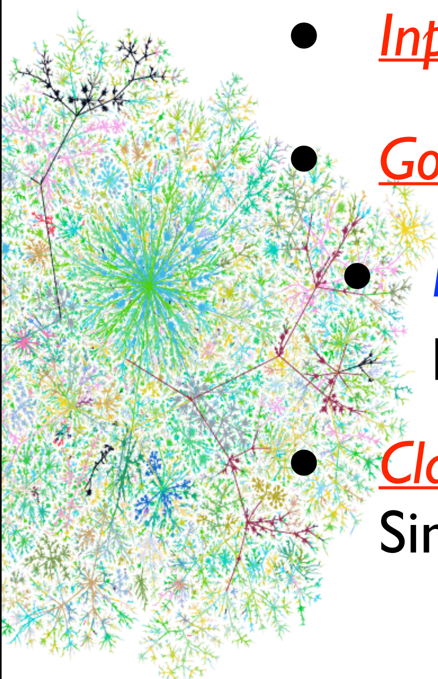


- *Input* Sets $S_1, S_2, \ldots, S_m \subseteq [n]$ and integer k
- *Goal* Pick k sets to maximize the size of the union of these sets

MaxCover Problem

- *Input* Sets $S_1, S_2, \ldots, S_m \subseteq [n]$ and integer k
- *Goal* Pick k sets to maximize the size of the union of these sets
    - *For example,* if $S_1=\{1,2,3\}$, $S_2=\{2,4,6\}$, $S_3=\{2,3,4\}$ and k=2 then picking S_1 and S_2 covers $\{1,2,3,4,6\}$.

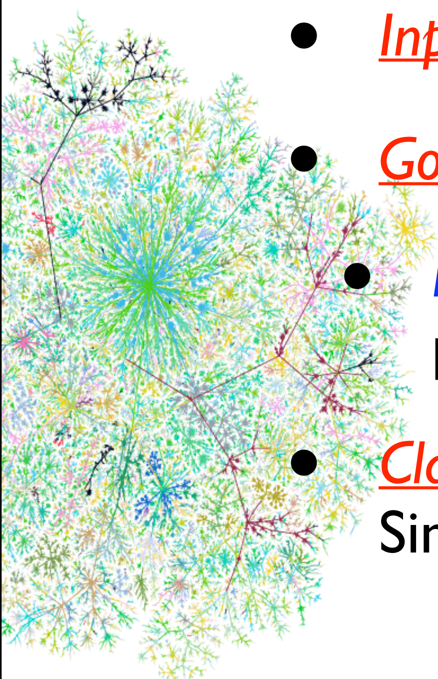
MaxCover Problem



- *Input* Sets $S_1, S_2, \ldots, S_m \subseteq [n]$ and integer k
- *Goal* Pick k sets to maximize the size of the union of these sets
  - *For example,* if $S_1=\{1,2,3\}$, $S_2=\{2,4,6\}$, $S_3=\{2,3,4\}$ and k=2 then picking S_1 and S_2 covers $\{1,2,3,4,6\}$.
- *Classic Results* Greedy algorithm is 1-1/e approx and is best possible. Simple example of sub-modular maximization optimization.

Feige [JACM 98]

MaxCover Problem



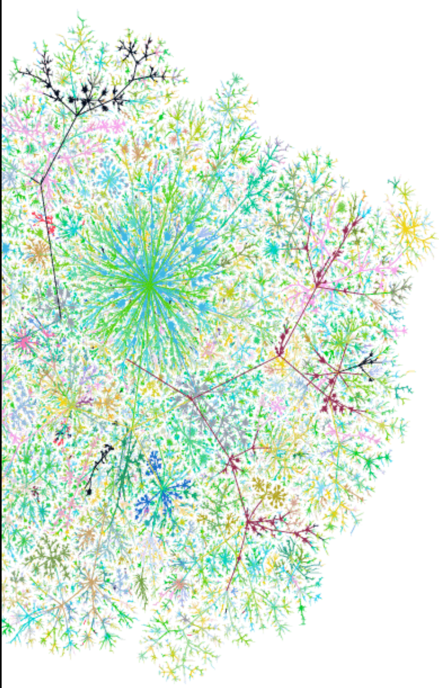
- *Input* Sets $S_1, S_2, \ldots, S_m \subseteq [n]$ and integer k
- *Goal* Pick k sets to maximize the size of the union of these sets
  - *For example,* if $S_1=\{1,2,3\}$, $S_2=\{2,4,6\}$, $S_3=\{2,3,4\}$ and k=2 then picking S_1 and S_2 covers $\{1,2,3,4,6\}$.
- *Classic Results* Greedy algorithm is 1-1/e approx and is best possible. Simple example of sub-modular maximization optimization.

Feige [JACM 98]

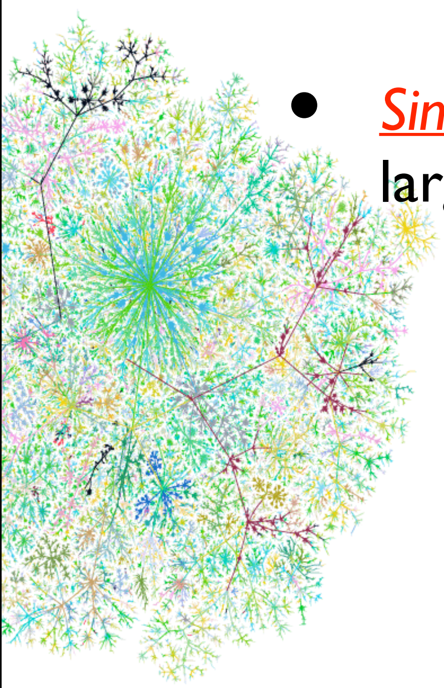
- *Data Streams* Growing body of work on MaxCover and SetCover considers input given as a stream of sets

Assadi et al. [STOC 16], *Warneke et al.* [ESA 23], *Indyk-Vakilian* [PODS 2019], *Yu-Yuan* [SDM 13]
Saha-Getoor [SDM 09], *McGregor-Vu* [ICDT 17], *Bateni et al.* [SPAA 17], *Assadi* [PODS 17]
Demaine et al. [DISC 14], *Indyk et al.* [APPROX 17], *Chakrabarti-Wirth* [SODA 16]
Assadi et al. [SODA 18], *Har-Peled et al.* [PODS 16] *etc.*

Talk Summary



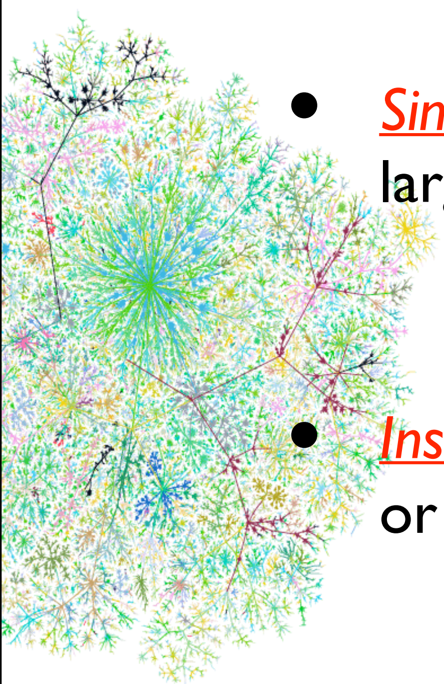
Talk Summary



- *Simplification* If sets are small, can throw out many sets. If sets are large, can subsample universe to ensure sets are small-ish.

Older results: [*Demaine et al.* DISC 14], [*McGregor, Vu.* Theory Comput. Syst. 19], [*McGregor et al.* ICDT 21]

Talk Summary



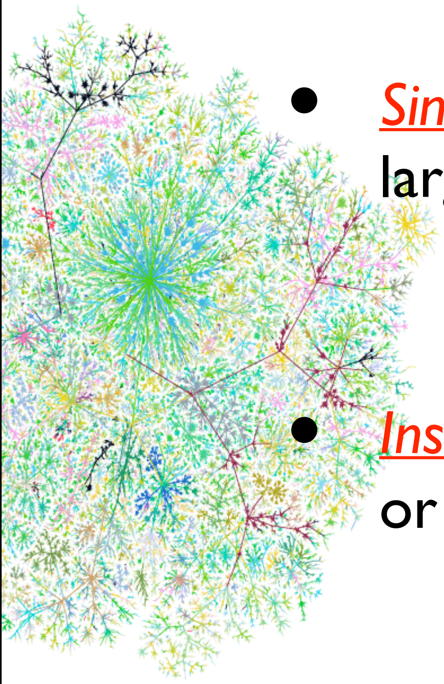
- *Simplification* If sets are small, can throw out many sets. If sets are large, can subsample universe to ensure sets are small-ish.

  Older results: [*Demaine et al.* DISC 14], [*McGregor, Vu.* Theory Comput. Syst. 19], [*McGregor et al.* ICDT 21]

- *Insert-Only Streams* $\tilde{O}_{\varepsilon}(k)$ space suffices for 1/2-ε approx (1-pass) or 1-1/e-ε (1-pass random order O(1/ε)-pass arbitrary order).

  Prev. best space bound for random order $\tilde{O}_{\varepsilon}(k^2)$ [*Warneke et al.* ESA 23]

Talk Summary



- *Simplification* If sets are small, can throw out many sets. If sets are large, can subsample universe to ensure sets are small-ish.

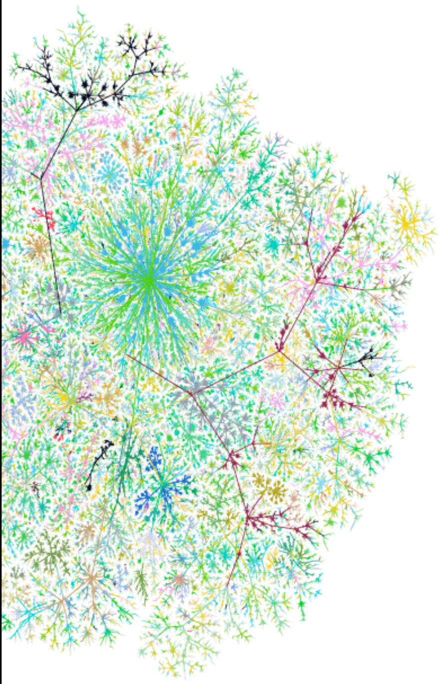
  Older results: [*Demaine et al.* DISC 14], [*McGregor, Vu.* Theory Comput. Syst. 19], [*McGregor et al.* ICDT 21]

- *Insert-Only Streams* $\tilde{O}_\varepsilon(k)$ space suffices for 1/2-ε approx (1-pass) or 1-1/e-ε (1-pass random order O(1/ε)-pass arbitrary order).

  Prev. best space bound for random order $\tilde{O}_\varepsilon(k^2)$ [*Warneke et al.* ESA 23]

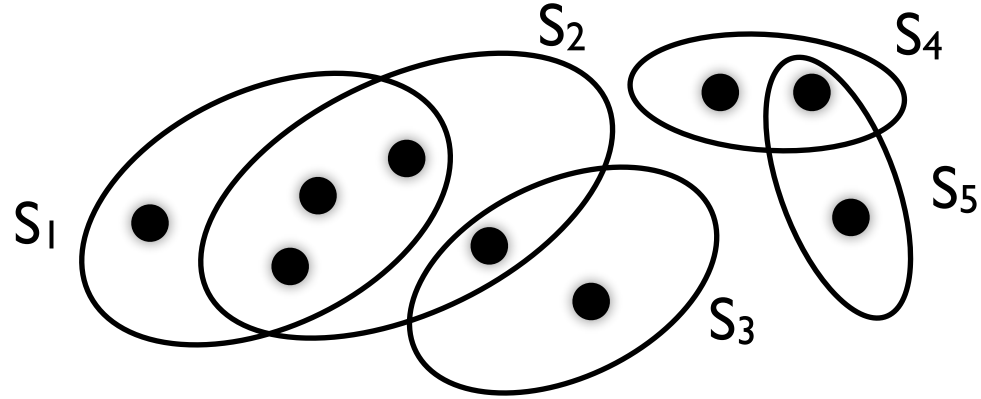
- *Dynamic Streams* $\tilde{O}(\varepsilon^{-2} k)$ space, $O(\log m+\varepsilon^{-1} \log m/\log \log m)$ passes suffice for 1-1/e-ε approx.

  [*Chakrabarti, McGregor, Wirth.* ESA 24]
  Prev. best space bound $\tilde{O}(n+\varepsilon^{-4} k)$ [*Assadi, Khanna.* SODA 18]

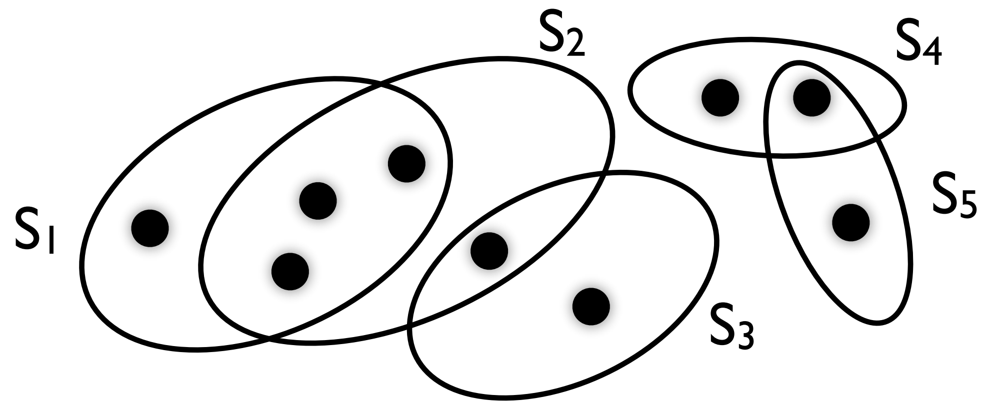


1: <u>**Simplification**</u>
2: **Insert Streams**
3: **Dynamic Streams**

Universe Sampling

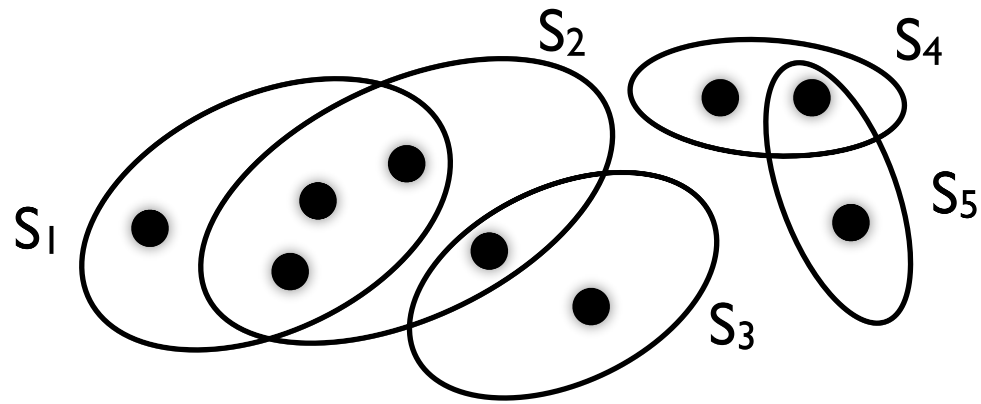


Universe Sampling



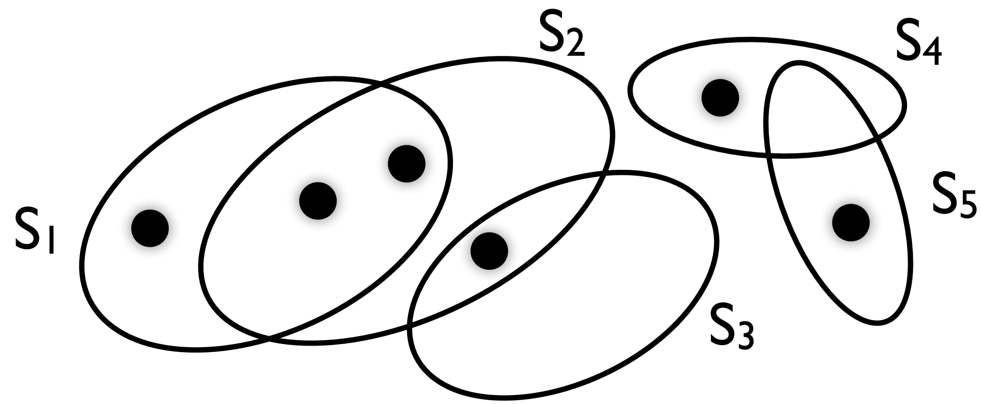
- Universe Sampling

Universe Sampling



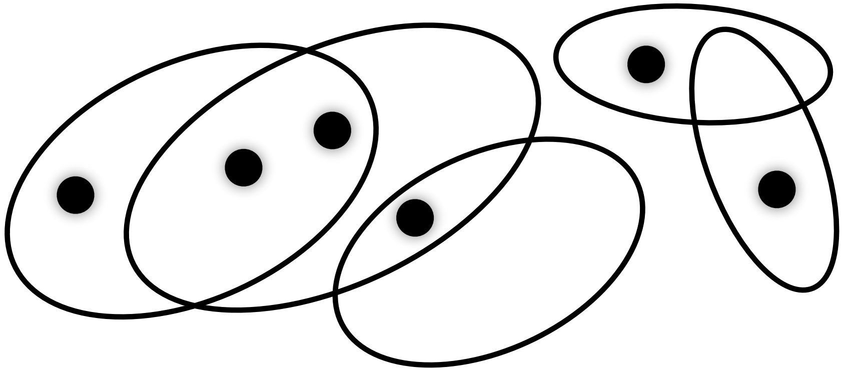
- <u>*Universe Sampling*</u>
    - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/\mathrm{OPT}$.

Universe Sampling



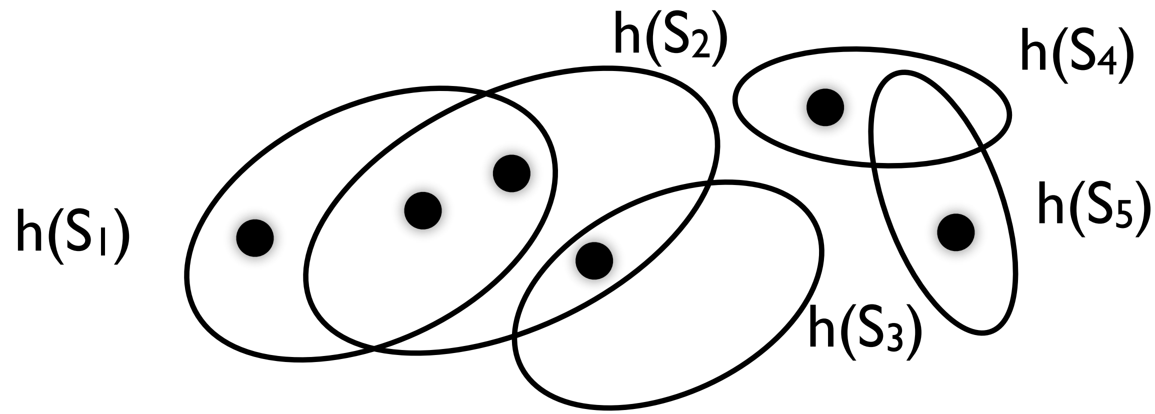
- *Universe Sampling*
    - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/OPT$.

Universe Sampling



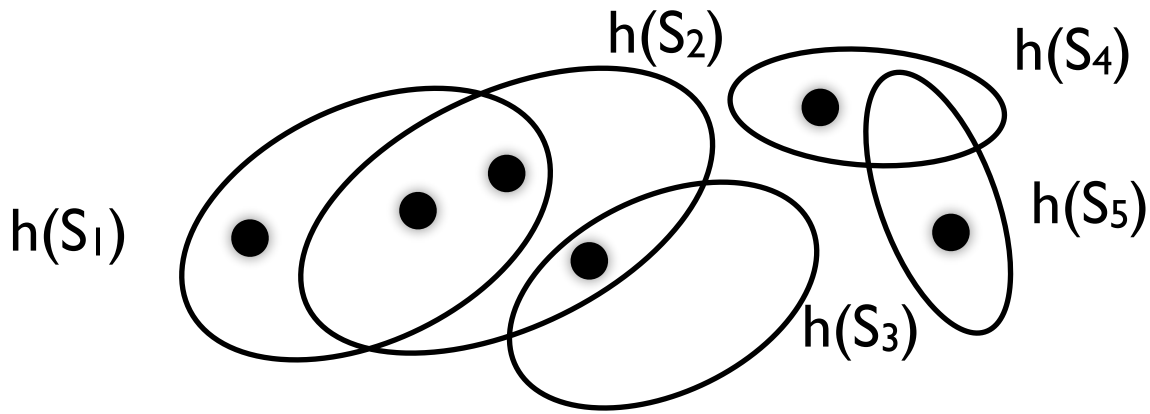
- *Universe Sampling*
    - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/OPT$.
    - For set S, let h(S) be the set of elements in S that were sampled.

Universe Sampling



- *Universe Sampling*
  - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/OPT$.
  - For set S, let h(S) be the set of elements in S that were sampled.

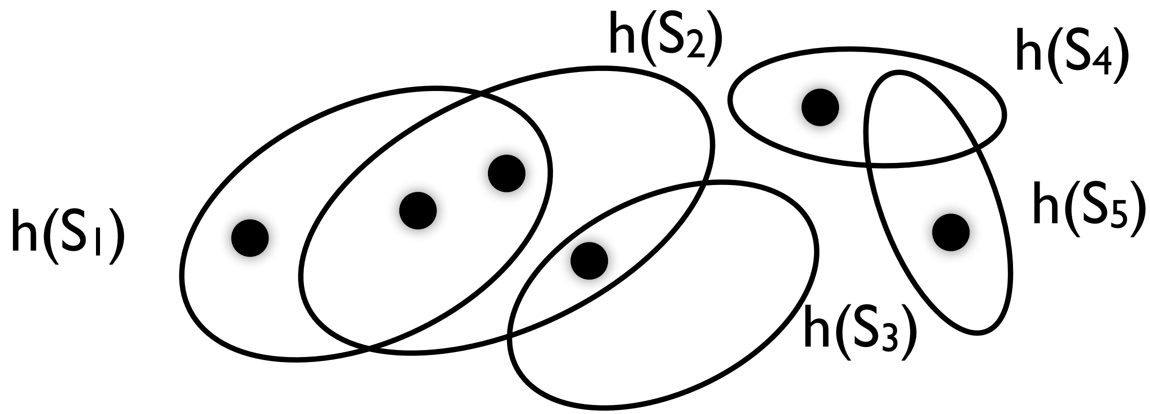
Universe Sampling



- <u>*Universe Sampling*</u>
  - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/\mathrm{OPT}$.
  - For set S, let h(S) be the set of elements in S that were sampled.
- <u>*Lemma*</u> Whp, for all $T_1, \ldots, T_k \in \{S_1, \ldots, S_m\}$ we have:

$$|h(T_1) \cup \ldots \cup h(T_k)|/p = |h(T_1) \cup \ldots \cup h(T_k)| \pm \varepsilon\, \mathrm{OPT}$$

Universe Sampling

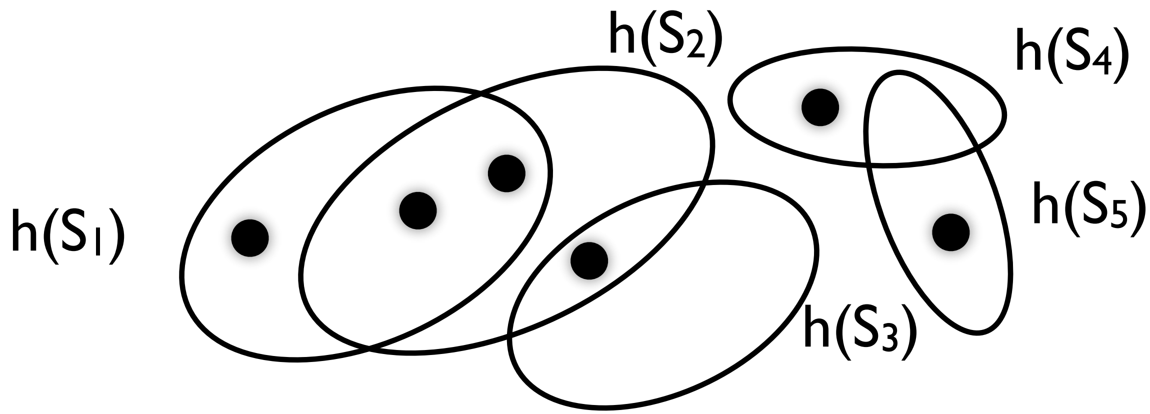


- *Universe Sampling*
  - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/\mathrm{OPT}$.
  - For set S, let h(S) be the set of elements in S that were sampled.
- *Lemma* Whp, for all $T_1, \ldots, T_k \in \{S_1, \ldots, S_m\}$ we have:

$$|h(T_1) \cup \ldots \cup h(T_k)|/p = |h(T_1) \cup \ldots \cup h(T_k)| \pm \varepsilon\, \mathrm{OPT}$$

- *Proof* Chernoff Bound + Union Bound over m^k collections of sets.

Universe Sampling

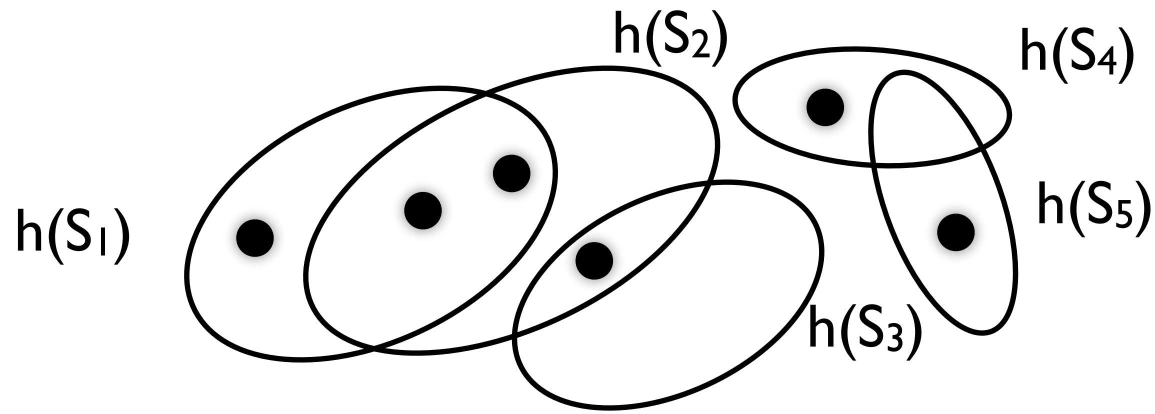


- *Universe Sampling*
  - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/OPT$.
  - For set S, let h(S) be the set of elements in S that were sampled.
- *Lemma* Whp, for all $T_1, \ldots, T_k \in \{S_1, \ldots, S_m\}$ we have:

$$|h(T_1) \cup \ldots \cup h(T_k)|/p = |h(T_1) \cup \ldots \cup h(T_k)| \pm \varepsilon\, OPT$$

- *Proof* Chernoff Bound + Union Bound over m^k collections of sets.
- α-approx on $h(S_1), \ldots, h(S_m)$ gives $(\alpha\text{-}\varepsilon)$-approx for original instance.

Universe Sampling

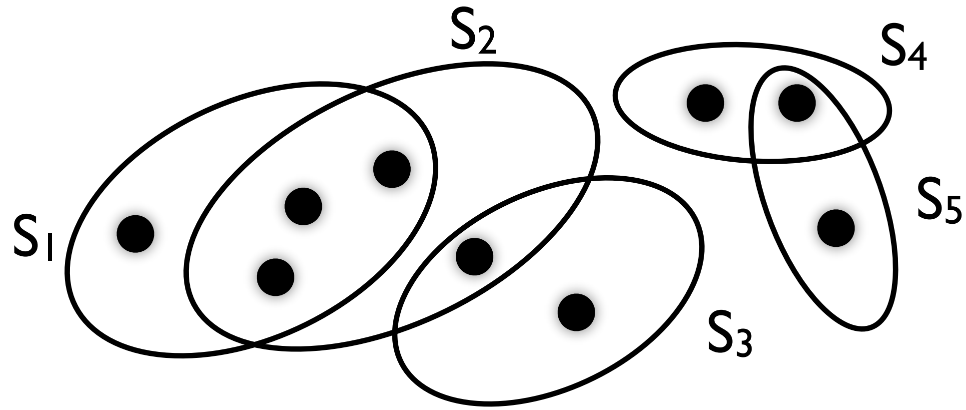


- *Universe Sampling*
  - Sample elements with probability $p \approx (k\varepsilon^{-2} \log m)/\mathrm{OPT}$.
  - For set S, let h(S) be the set of elements in S that were sampled.
- *Lemma* Whp, for all $T_1, \ldots, T_k \in \{S_1, \ldots, S_m\}$ we have:

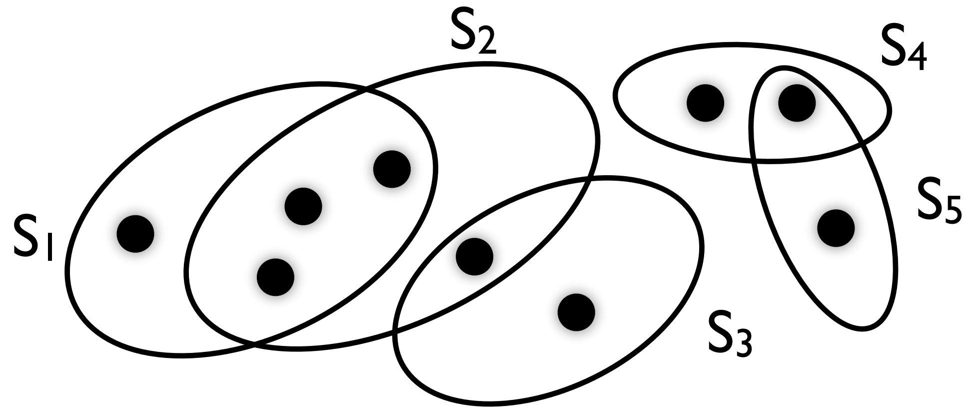
$$|h(T_1)\cup\ldots\cup h(T_k)|/p = |h(T_1)\cup\ldots\cup h(T_k)| \pm \varepsilon\,\mathrm{OPT}$$

- *Proof* Chernoff Bound + Union Bound over m^k collections of sets.
- α-approx on $h(S_1),\ldots, h(S_m)$ gives $(\alpha\text{-}\varepsilon)$-approx for original instance.
- Optimum solution for $h(S_1),\ldots, h(S_m)$ covers $O(k\varepsilon^{-2} \log m)$ elts.

Simple Set Sampling (Kernelization)

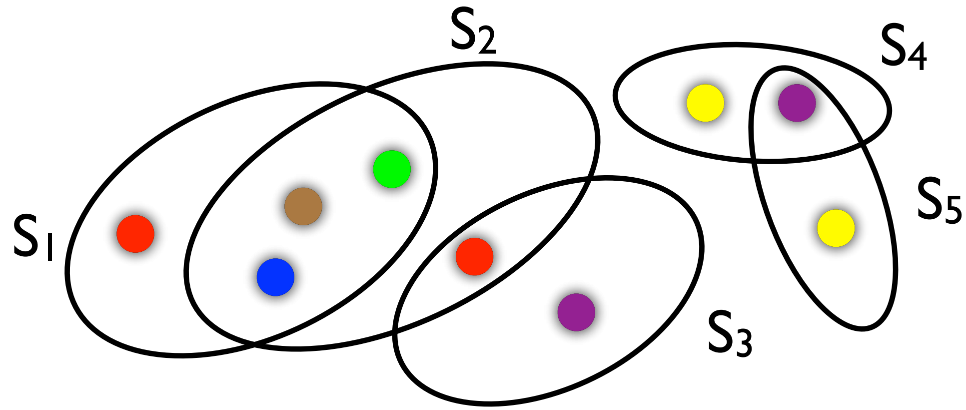


Simple Set Sampling (Kernelization)



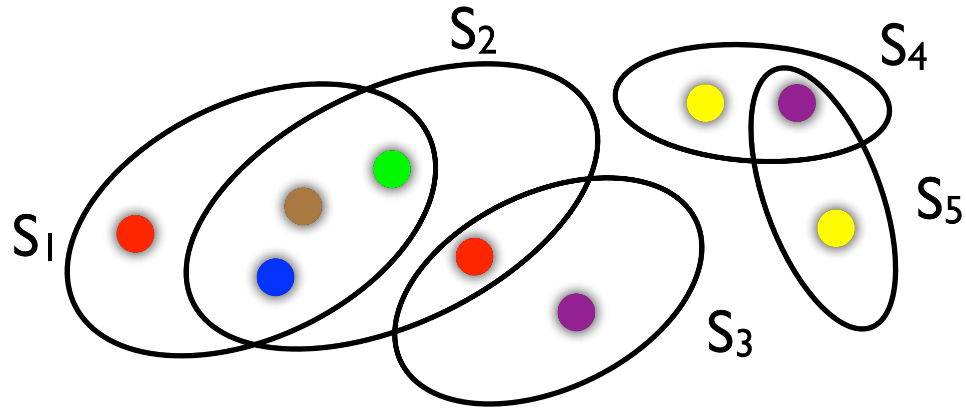
- *Set Sampling*
    - Let color:[n]→[c] be a random hash function

Simple Set Sampling (Kernelization)



- *Set Sampling*
  - Let color:[n]→[c] be a random hash function

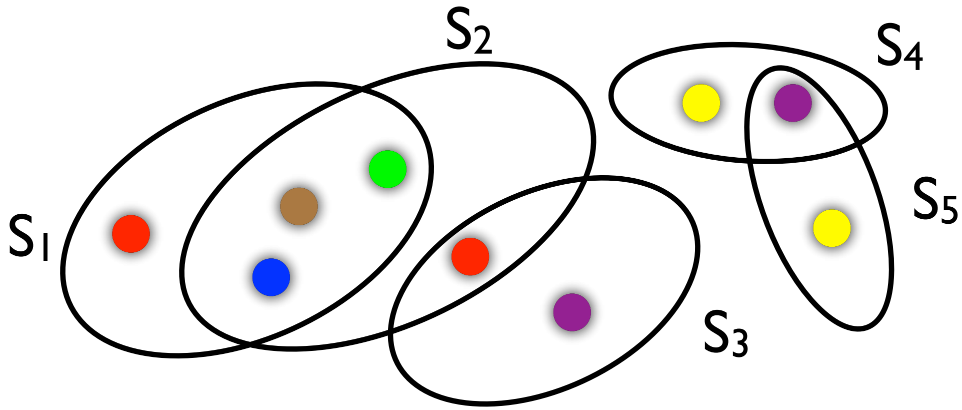
Simple Set Sampling (Kernelization)



- *Set Sampling*
  - Let color:[n]→[c] be a random hash function
  - Only keep sets with distinct color(S)={color(i): i ∈ S}

Simple Set Sampling (Kernelization)

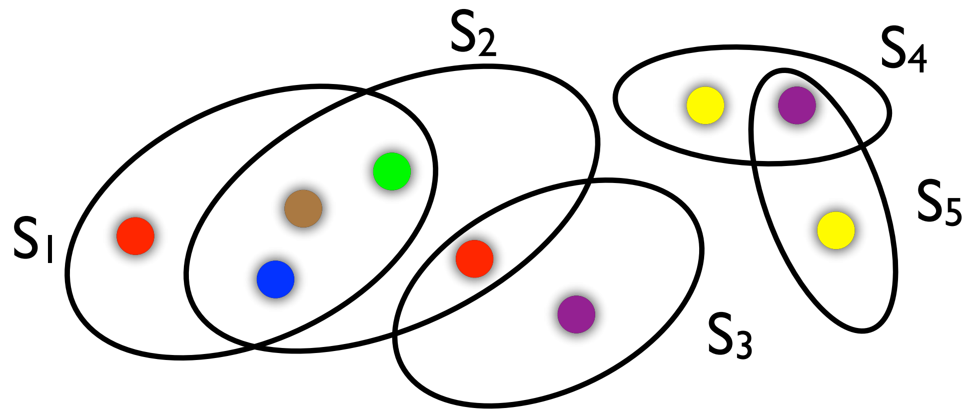
Remove one of $\{S_1, S_2\}$ and one of $\{S_4,S_5\}$.



- *Set Sampling*
  - Let color:[n]→[c] be a random hash function
  - Only keep sets with distinct color(S)={color(i): i ∈ S}

Simple Set Sampling (Kernelization)

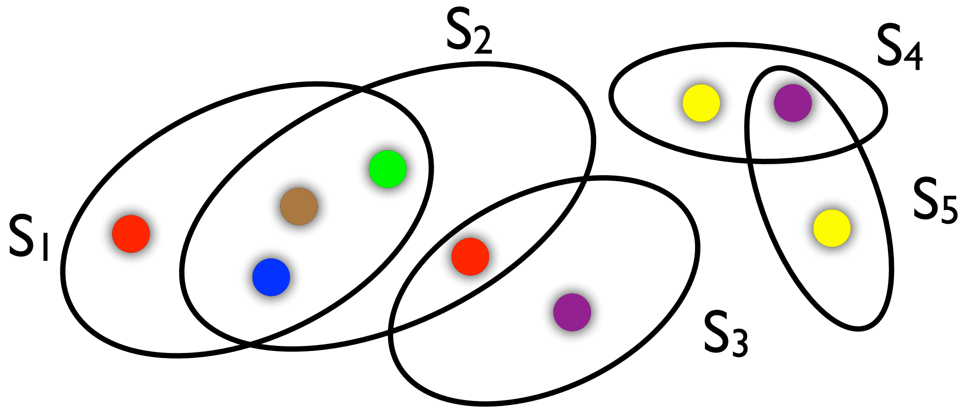
Remove one of $\{S_1, S_2\}$ and one of $\{S_4, S_5\}$.



- *Set Sampling*
  - Let $\text{color}:[n] \to [c]$ be a random hash function
  - Only keep sets with distinct $\text{color}(S)=\{\text{color}(i): i \in S\}$
- *Warm-up* Let r be $\max_i |S_i|$. If $c \approx (kr)^2$ there's $\le (kr)^{2r}$ retained sets and these include an opt solution with good probability.

Simple Set Sampling (Kernelization)

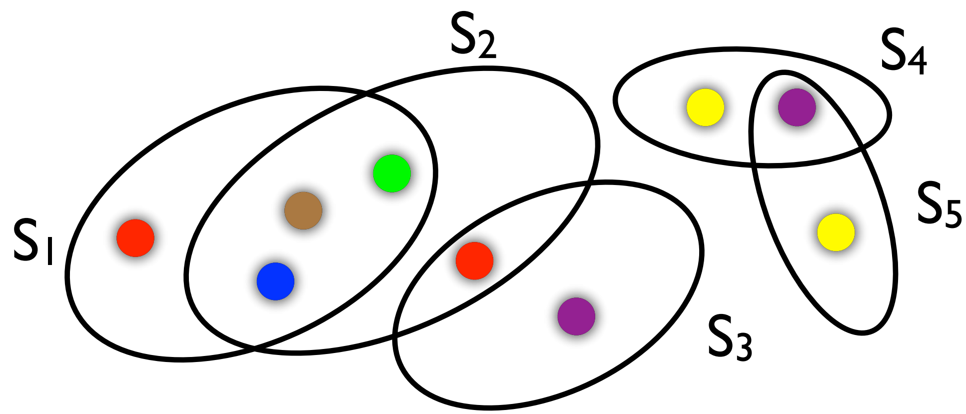
Remove one of $\{S_1, S_2\}$ and one of $\{S_4, S_5\}$.



- *Set Sampling*
  - Let color:$[n] \to [c]$ be a random hash function
  - Only keep sets with distinct color(S)=$\{$color(i): $i \in S\}$
- *Warm-up* Let r be $\max_i |S_i|$. If $c \approx (kr)^2$ there's $\leq (kr)^{2r}$ retained sets and these include an opt solution with good probability.
  - Let $O_1, \ldots, O_k$ be an optimum solution. Covers $\leq$ kr elements.

Simple Set Sampling (Kernelization)

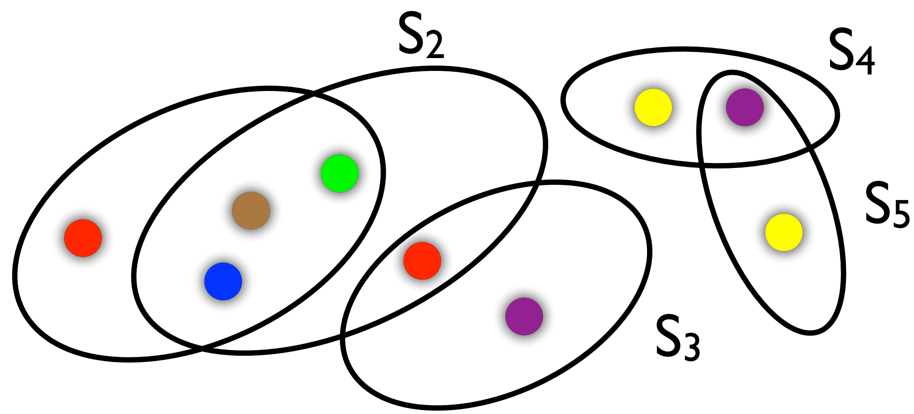
Remove one of $\{S_1, S_2\}$ and one of $\{S_4, S_5\}$.



- *Set Sampling*
  - Let color:$[n] \to [c]$ be a random hash function
  - Only keep sets with distinct color(S)=$\{$color(i): $i \in S\}$
- *Warm-up* Let r be $\max_i |S_i|$. If $c \approx (kr)^2$ there's $\leq (kr)^{2r}$ retained sets and these include an opt solution with good probability.
  - Let $O_1, \ldots, O_k$ be an optimum solution. Covers $\leq kr$ elements.
  - By Birthday-Paradox, $|O_1 \cup \ldots \cup O_k| = |\text{color}(O_1 \cup \ldots \cup O_k)|$

Simple Set Sampling (Kernelization)

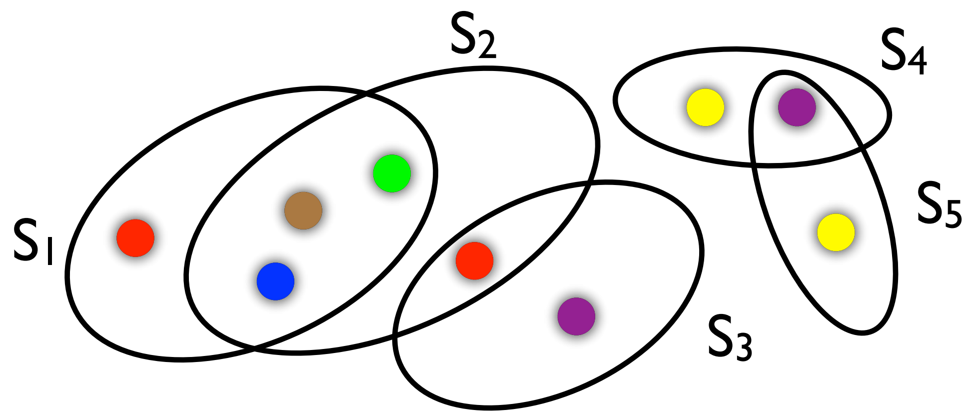
Remove one of $\{S_1, S_2\}$ and one of $\{S_4, S_5\}$.



- *Set Sampling*
  - Let color:$[n] \rightarrow [c]$ be a random hash function
  - Only keep sets with distinct color(S)={color(i): $i \in S$}
- *Warm-up* Let r be $\max_i |S_i|$. If $c \approx (kr)^2$ there's $\leq (kr)^{2r}$ retained sets and these include an opt solution with good probability.
  - Let $O_1, \ldots, O_k$ be an optimum solution. Covers $\leq kr$ elements.
  - By Birthday-Paradox, $|O_1 \cup \ldots \cup O_k| = |\text{color}(O_1 \cup \ldots \cup O_k)|$
  - If R_i is set retained with color(R_i)=color(O_i) then

Simple Set Sampling (Kernelization)

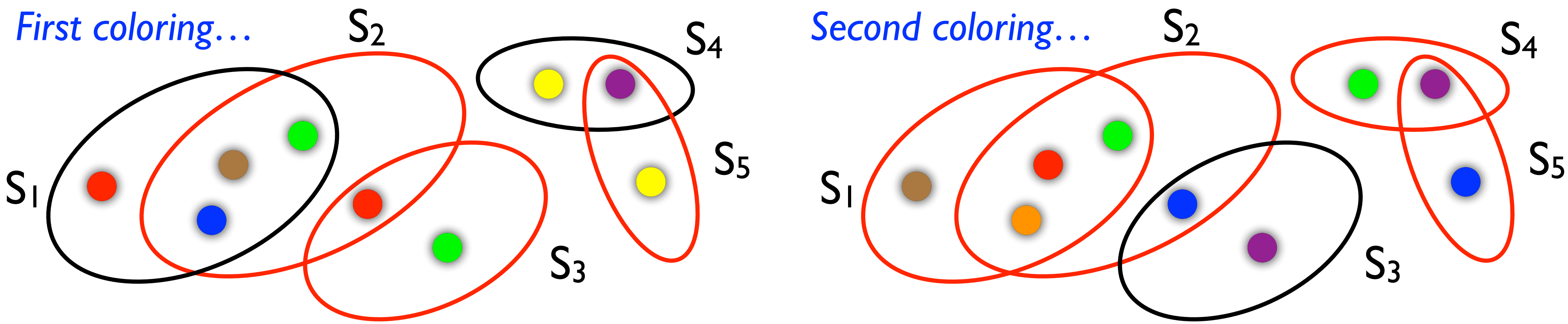
Remove one of $\{S_1, S_2\}$ and one of $\{S_4, S_5\}$.



- *Set Sampling*
  - Let color:$[n] \to [c]$ be a random hash function
  - Only keep sets with distinct color(S)={color(i): $i \in S$}
- *Warm-up* Let r be $\max_i |S_i|$. If $c \approx (kr)^2$ there's $\leq (kr)^{2r}$ retained sets and these include an opt solution with good probability.
  - Let $O_1, \ldots, O_k$ be an optimum solution. Covers $\leq kr$ elements.
  - By Birthday-Paradox, $|O_1 \cup \ldots \cup O_k| = |\text{color}(O_1 \cup \ldots \cup O_k)|$
  - If R_i is set retained with color(R_i)=color(O_i) then

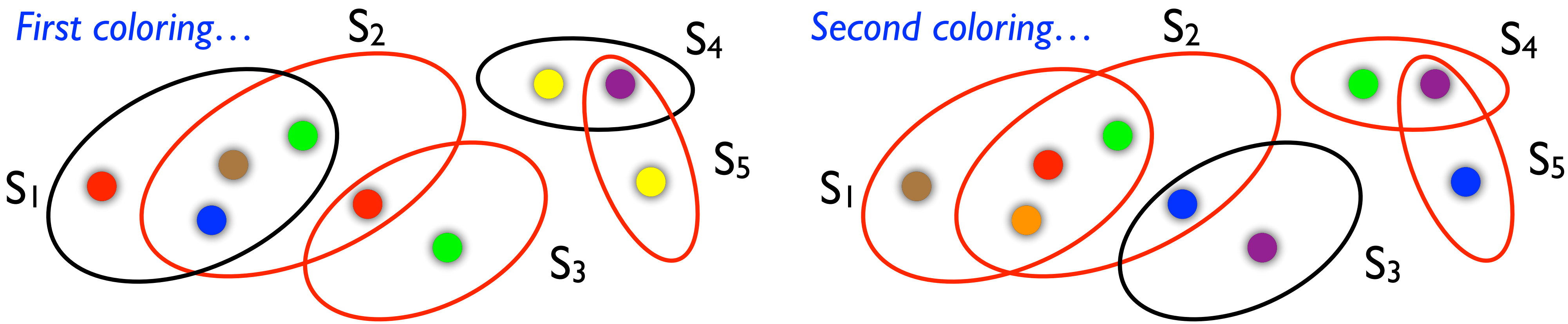
$$|\text{color}(O_1 \cup \ldots \cup O_k)| = |\text{color}(R_1 \cup \ldots \cup R_k)| \leq |R_1 \cup \ldots \cup R_k|$$

Superior Set Sampling



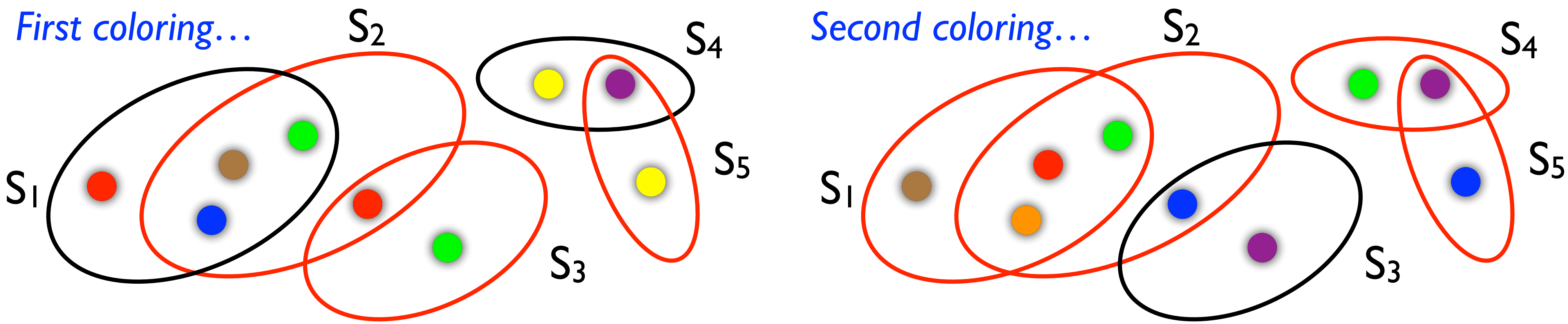
- <u>*Superior Set Sampling*</u>
    - Let $\mathrm{color}_1, \ldots, \mathrm{color}_t : [n] \rightarrow [c]$ be t random hash functions

Superior Set Sampling



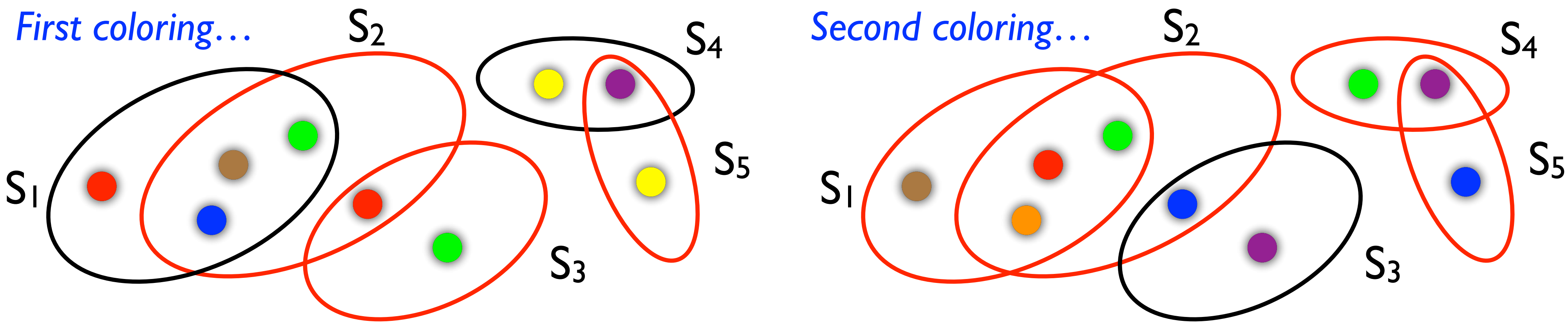
- <u>*Superior Set Sampling*</u>
  - Let $color_1, \ldots, color_t : [n] \rightarrow [c]$ be t random hash functions
  - For $j \in [t]$, retain maximal collection of sets with distinct $color_j(\cdot)$

Superior Set Sampling



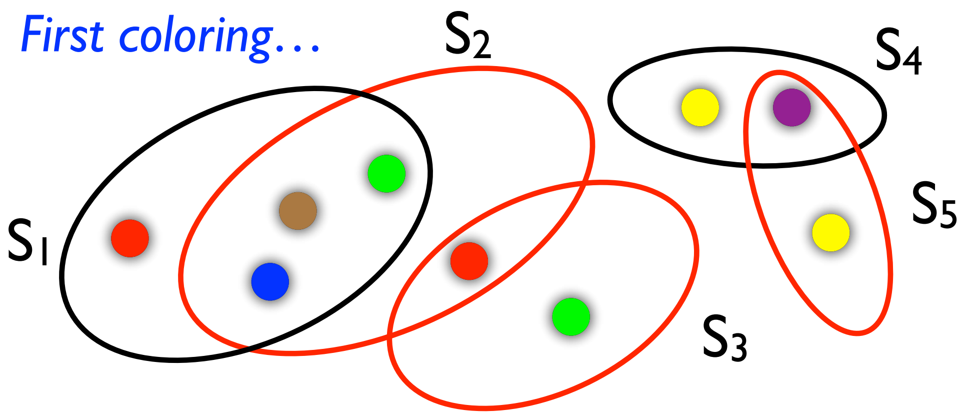
- *Superior Set Sampling*
  - Let $\text{color}_1, \ldots, \text{color}_t : [n] \rightarrow [c]$ be t random hash functions
  - For $j \in [t]$, retain maximal collection of sets with distinct $\text{color}_j(\cdot)$
- *Theorem* If $c \approx kr^2$ and $t \approx \log k$ there are $\approx (kr^2)^r$ retained sets and these include an opt solution whp. *McGregor-Tench-Vu* [ICDT 21]

Superior Set Sampling



- *Superior Set Sampling*
  - Let $\text{color}_1, \ldots, \text{color}_t : [n] \rightarrow [c]$ be t random hash functions
  - For $j \in [t]$, retain maximal collection of sets with distinct $\text{color}_j(\cdot)$
- *Theorem* If $c \approx kr^2$ and $t \approx \log k$ there are $\approx (kr^2)^r$ retained sets and these include an opt solution whp. *McGregor-Tench-Vu* [ICDT 21]
- *Proof Approach* Fix opt solution. In each coloring, for each opt set with constant prob we sample that set or a set just as good…

Superior Set Sampling: Proof Idea

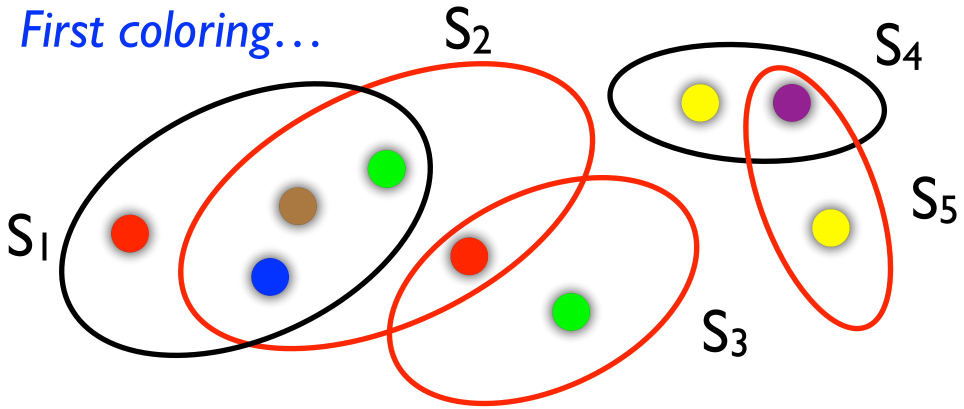


$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

- Let O_i be opt set and R_i set retained with color(R_i)=color(O_i).

Superior Set Sampling: Proof Idea

First coloring…

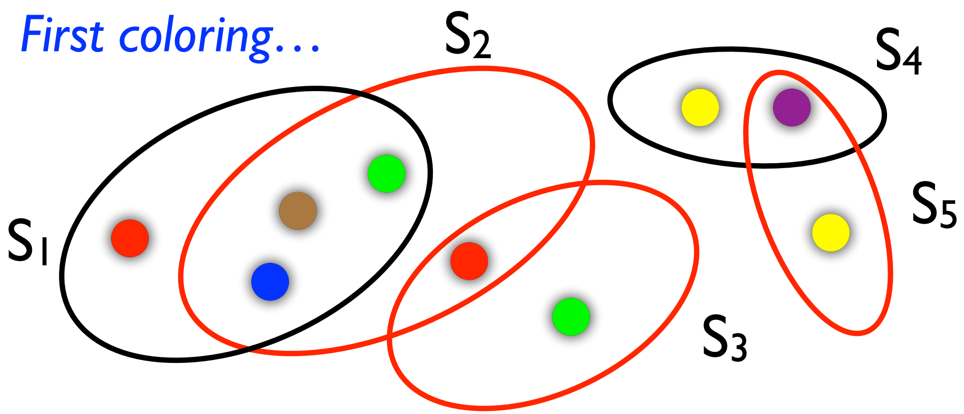


$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

- Let O_i be opt set and R_i set retained with $\text{color}(R_i)=\text{color}(O_i)$.
- Say O_i is color-full if

Superior Set Sampling: Proof Idea

First coloring…



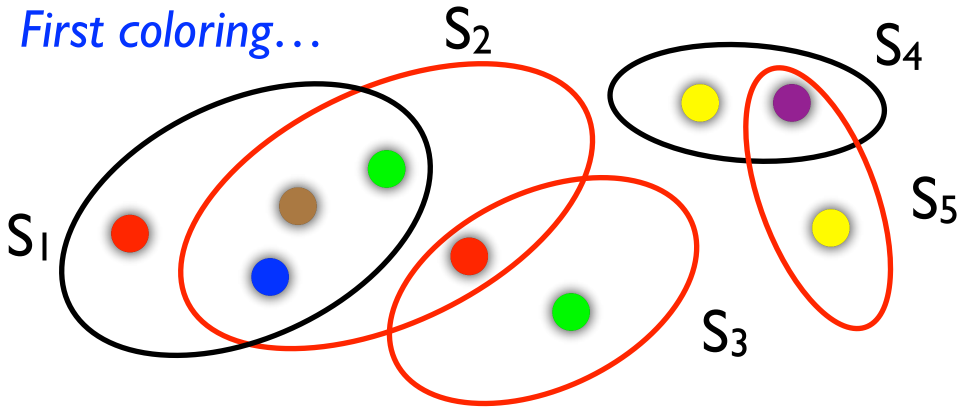
$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

- Let O_i be opt set and R_i set retained with color(R_i)=color(O_i).
- Say O_i is color-full if

$$|\text{color}(O_i)| = |O_i| \text{ and } \text{color}(O_i) \cap \text{color}(O_1 \cup \ldots \cup O_k \setminus O_i) = \varnothing$$

Superior Set Sampling: Proof Idea

First coloring…



$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

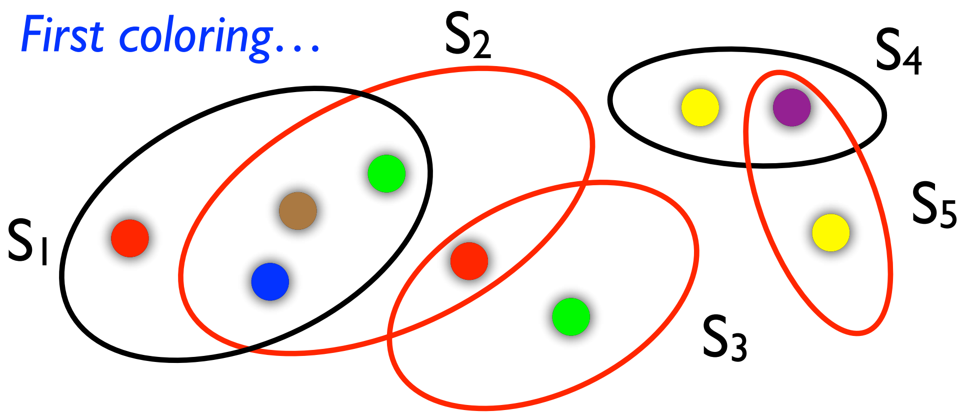
- Let O_i be opt set and R_i set retained with $\text{color}(R_i)=\text{color}(O_i)$.
- Say O_i is color-full if

$$|\text{color}(O_i)| = |O_i| \text{ and } \text{color}(O_i) \cap \text{color}(O_1 \cup \ldots \cup O_k \setminus O_i) = \varnothing$$

- If O_i is color-full, replace $O_i \leftarrow R_i$. Note $|O_1 \cup \ldots \cup O_k|$ unchanged.

Superior Set Sampling: Proof Idea

First coloring…



$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

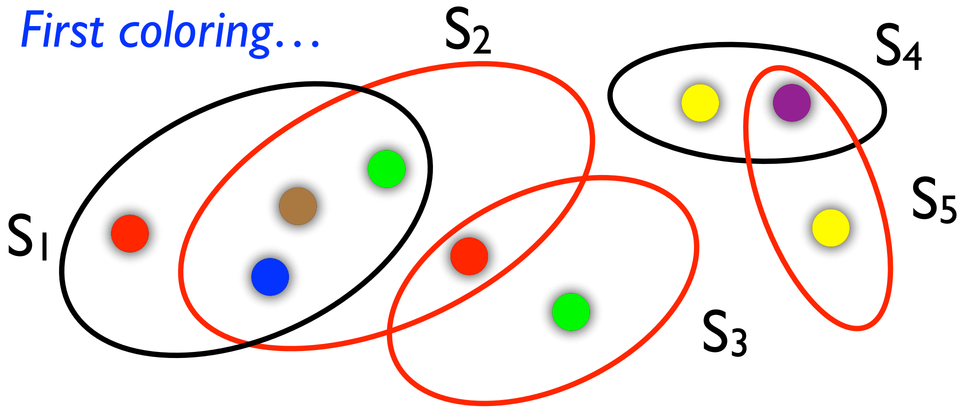
- Let O_i be opt set and R_i set retained with $\text{color}(R_i)=\text{color}(O_i)$.
- Say O_i is color-full if

  $$|\text{color}(O_i)| = |O_i| \text{ and } \text{color}(O_i) \cap \text{color}(O_1 \cup \ldots \cup O_k \setminus O_i) = \varnothing$$

- If O_i is color-full, replace $O_i \leftarrow R_i$. Note $|O_1 \cup \ldots \cup O_k|$ unchanged.
- $\Pr[O_i$ is color-full$] \geq 1-r^2k/c \geq 1/2$ and repeating log k times ensures we find k sets covering an optimum number of elts.

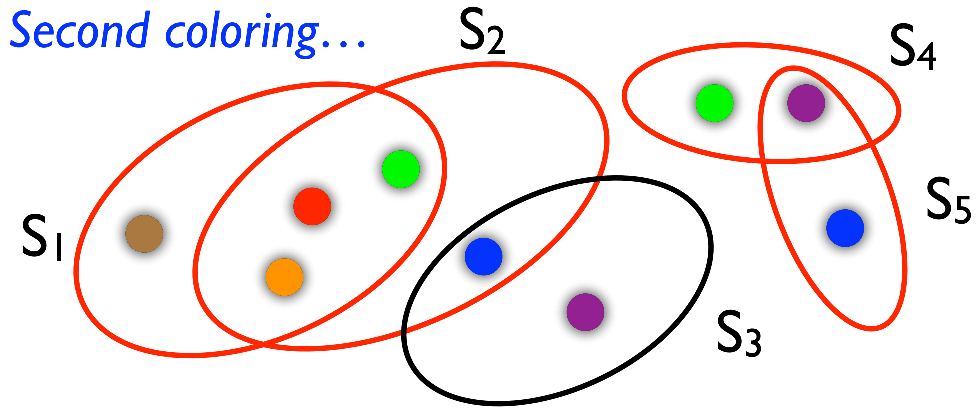
Superior Set Sampling: Proof Idea

First coloring…



$O_1=S_1$, $O_2=S_3$, $O_3=S_4$
$R_1=S_2$, $R_2=S_3$, $R_3=S_5$

Second coloring…

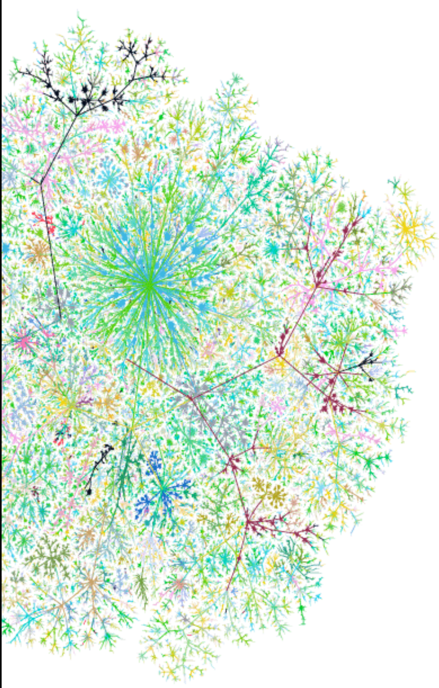


$O_1=S_1$, $O_2=S_3$, $O_3=S_5$
$R_1=S_1$, $R_2=S_5$, $R_3=S_5$

- Let O_i be opt set and R_i set retained with $\text{color}(R_i)=\text{color}(O_i)$.
- Say O_i is color-full if

  $$|\text{color}(O_i)| = |O_i| \text{ and } \text{color}(O_i) \cap \text{color}(O_1 \cup \ldots \cup O_k \setminus O_i) = \varnothing$$

- If O_i is color-full, replace $O_i \leftarrow R_i$. Note $|O_1 \cup \ldots \cup O_k|$ unchanged.
- $\Pr[O_i$ is color-full$] \geq 1-r^2k/c \geq 1/2$ and repeating log k times ensures we find k sets covering an optimum number of elts.



1: Simplification

2: Insert Streams

3: Dynamic Streams

One Pass Algorithm

- **Initialize:** Let $C=\emptyset$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

One Pass Algorithm

- **Initialize:** Let $C=\emptyset$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

- *Can 0.5-Approximation if we know value of optimum*

One Pass Algorithm

- **Initialize:** Let $C=\emptyset$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

- *Can 0.5-Approximation if we know value of optimum*
  - If k sets get added, we've covered ≥ OPT/2 elts

One Pass Algorithm

- **Initialize:** Let $C=\varnothing$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

- *Can 0.5-Approximation if we know value of optimum*
  - If k sets get added, we've covered ≥ OPT/2 elts
  - If <k sets get added, adding optimal sets adds ≤ OPT/2 new elts so must have already covered ≥ OPT/2 elts.

cf. "Sieve Streaming" [Badanidiyuru et al. KDD 14]

One Pass Algorithm

- **Initialize:** Let $C=\emptyset$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

- *Can 0.5-Approximation if we know value of optimum*
  - If k sets get added, we've covered ≥ OPT/2 elts
  - If <k sets get added, adding optimal sets adds ≤ OPT/2 new elts so must have already covered ≥ OPT/2 elts.

cf. "Sieve Streaming" [*Badanidiyuru et al.* KDD 14]

- *Theorem* 1 pass, $\tilde{O}(\varepsilon^{-3} k)$ space, $1/2-\varepsilon$ approx algorithm.

One Pass Algorithm

- **Initialize:** Let $C = \emptyset$ be covered elements
- **Pass:** Add any set that covers $\geq OPT/(2k)$ new elements.

- *Can 0.5-Approximation if we know value of optimum*
  - If k sets get added, we've covered $\geq OPT/2$ elts
  - If <k sets get added, adding optimal sets adds $\leq OPT/2$ new elts so must have already covered $\geq OPT/2$ elts.

cf. "Sieve Streaming" [*Badanidiyuru et al.* KDD 14]

- *Theorem* 1 pass, $\tilde{O}(\varepsilon^{-3} k)$ space, $1/2-\varepsilon$ approx algorithm.
- *Proof* Combine sub-sampling approach with threshold algorithm. Extra $\tilde{O}(\varepsilon^{-1})$ arises from having to guess OPT up to $1+\varepsilon$ factor.

Multi-Pass Algorithm

- **In pass $p = 1, \ldots, O(1/\varepsilon)$:**
  - Add any set that covers $\geq$ OPT Θ_p/k new elements where threshold Θ_p decreases as $1, 1/(1+\varepsilon), 1/(1+\varepsilon)^2 \ldots 1/(2e)$

Multi-Pass Algorithm

- In pass $p = 1, \ldots, O(1/\varepsilon)$:
  - Add any set that covers $\geq \mathrm{OPT}\ \Theta_p/k$ new elements where threshold Θ_p decreases as $1, 1/(1+\varepsilon), 1/(1+\varepsilon)^2 \ldots\ 1/(2e)$

- *Theorem* $O(\varepsilon^{-1})$ pass $1-1/e-\varepsilon$ approx using $\tilde{O}(\varepsilon^{-2}\, k)$ space.

Multi-Pass Algorithm

- In pass $p = 1, \ldots, O(1/\varepsilon)$:
  - Add any set that covers $\geq OPT\ \Theta_p/k$ new elements where threshold Θ_p decreases as $1, 1/(1+\varepsilon), 1/(1+\varepsilon)^2 \ldots\ 1/(2e)$

- *Theorem* $O(\varepsilon^{-1})$ pass $1-1/e-\varepsilon$ approx using $\tilde{O}(\varepsilon^{-2}\ k)$ space.
- *Proof* Approx factor follows from analysis of greedy algorithm. Universe subsampling and fact we only need 2-approx of OPT.

Random Order Algorithm

- Recall, function $f: 2^{[m]} \to \mathbb{R}$ is sub-modular if for $X \subset Y \subseteq [m]$, $i \notin Y$

  $f(X+i)-f(X) \geq f(Y+i)-f(Y)$

Random Order Algorithm

- Recall, function $f: 2^{[m]} \to \mathbb{R}$ is sub-modular if for $X \subset Y \subseteq [m]$, $i \notin Y$

  $f(X+i)-f(X) \geq f(Y+i)-f(Y)$

  e.g., $f(X)=|\cup_{i\in X} S_i|$ where $S_1, \ldots, S_m$ are sets

Random Order Algorithm

- Recall, function $f: 2^{[m]} \rightarrow \mathbb{R}$ is sub-modular if for $X \subset Y \subseteq [m]$, $i \notin Y$

  $f(X+i)-f(X) \geq f(Y+i)-f(Y)$

  e.g., $f(X)=|\cup_{i \in X} S_i|$ where $S_1, \ldots, S_m$ are sets

- Work on sub-modular maximization in streams assumes stream is a permutation of $[m]$ and algorithm has oracle access to f.

  Norouzi-Fard et al. [ICML 18], Agrawal et al. [ITCS 19], Feldman et al. [STOC 20]

Random Order Algorithm

- Recall, function $f: 2^{[m]} \to \mathbb{R}$ is sub-modular if for $X \subset Y \subseteq [m]$, $i \notin Y$

  $f(X+i)-f(X) \geq f(Y+i)-f(Y)$

  e.g., $f(X)=|\cup_{i \in X} S_i|$ where $S_1, \ldots, S_m$ are sets

- Work on sub-modular maximization in streams assumes stream is a permutation of [m] and algorithm has oracle access to f.

  Norouzi-Fard et al. [ICML 18], Agrawal et al. [ITCS 19], Feldman et al. [STOC 20]

- *Corollary* Sub-modular results and universe sampling gives 1-pass, $\tilde{O}_\varepsilon(k^2)$-space, $1-1/e-\varepsilon$ approx. for random order streams.

Random Order Algorithm

- Recall, function $f: 2^{[m]} \to \mathbb{R}$ is sub-modular if for $X \subset Y \subseteq [m]$, $i \notin Y$

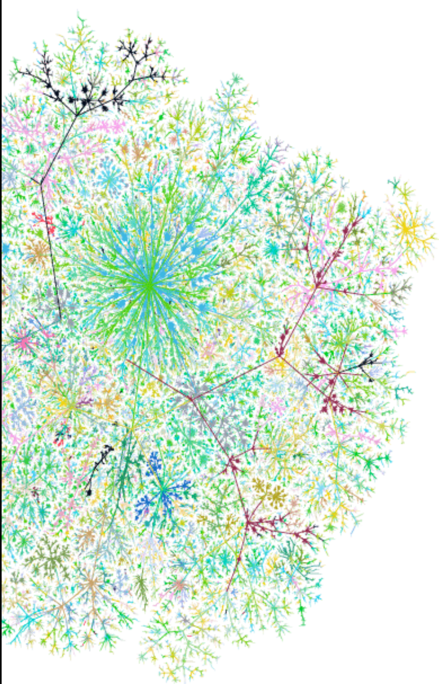
  $f(X+i)-f(X) \geq f(Y+i)-f(Y)$

  e.g., $f(X)=|\cup_{i\in X} S_i|$ where $S_1, \ldots, S_m$ are sets

- Work on sub-modular maximization in streams assumes stream is a permutation of [m] and algorithm has oracle access to f.
  Norouzi-Fard et al. [ICML 18], Agrawal et al. [ITCS 19], Feldman et al. [STOC 20]

- *Corollary* Sub-modular results and universe sampling gives 1-pass, $\tilde{O}_\varepsilon(k^2)$-space, $1-1/e-\varepsilon$ approx. for random order streams.

- *With more work…* Can reduce space dependence to linear in k.
  Chakrabarti, McGregor, Wirth [ESA 24]



1: **Simplification**
2: **Insert Streams**
3: **<u>Dynamic Streams</u>**

Dynamic Steam Algorithm: Warm-Up

Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.

Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

- **Initialize:** Let $C=\varnothing$ be covered elements
- **Until k sets added or no remaining sets are useful:**
  - During a pass: Sample k sets amongst the useful sets
  - At end of pass: Add sampled set that remain useful

Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

- **Initialize:** Let C=∅ be covered elements
- **Until k sets added or no remaining sets are useful:**
  - During a pass: Sample k sets amongst the useful sets
  - At end of pass: Add sampled set that remain useful

- *1/2 Approx* Carries over from insert-only analysis.

Dynamic Steam Algorithm: Warm-Up

- Definition Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

  - **Initialize:** Let C=∅ be covered elements
  - **Until k sets added or no remaining sets are useful:**
    - During a pass: Sample k sets amongst the useful sets
    - At end of pass: Add sampled set that remain useful

- 1/2 Approx Carries over from insert-only analysis.
- Space Can sample k sets in $\approx k \max_i |S_i \setminus C| = \tilde{O}_\varepsilon(k^2)$ space

Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

  - **Initialize:** Let $C=\varnothing$ be covered elements
  - **Until k sets added or no remaining sets are useful:**
    - During a pass: Sample k sets amongst the useful sets
    - At end of pass: Add sampled set that remain useful

- *1/2 Approx* Carries over from insert-only analysis.
- *Space* Can sample k sets in $\approx k \max_i |S_i \setminus C| = \tilde{O}_\varepsilon(k^2)$ space
- *Lemma* In each pass, number of remaining useful sets halves or we add k/3 new sets. So O(log m) passes.

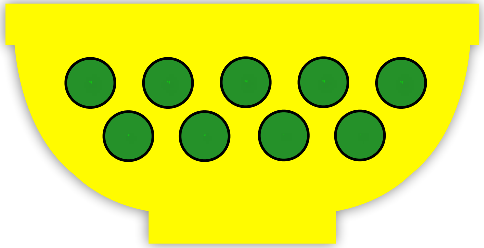
Dynamic Steam Algorithm: Warm-Up

- *Definition* Set is useful if it covers ≥ OPT/(2k) new elts.
- If S and T are useful, T may no longer be useful once S is added.

  - **Initialize:** Let $C=\varnothing$ be covered elements
  - **Until k sets added or no remaining sets are useful:**
    - During a pass: Sample k sets amongst the useful sets
    - At end of pass: Add sampled set that remain useful

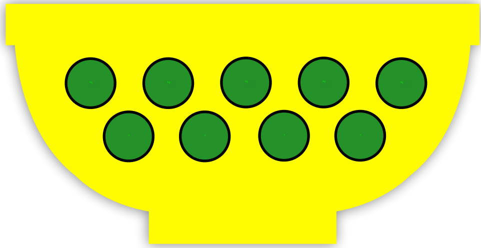
- *1/2 Approx* Carries over from insert-only analysis.
- *Space* Can sample k sets in $\approx k \max_i |S_i \setminus C| = \tilde{O}_\varepsilon(k^2)$ space
- *Lemma* In each pass, number of remaining useful sets halves or we add k/3 new sets. So O(log m) passes.
- A more careful analysis, gives O(log m/log log m) passes.

Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets

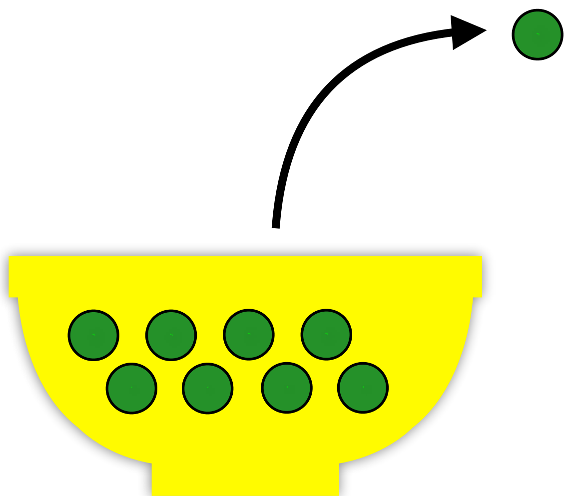
Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn $1 but an arbitrary number of other balls in urn become useless

* *Although principle of deferred decision means this doesn't really matter.*

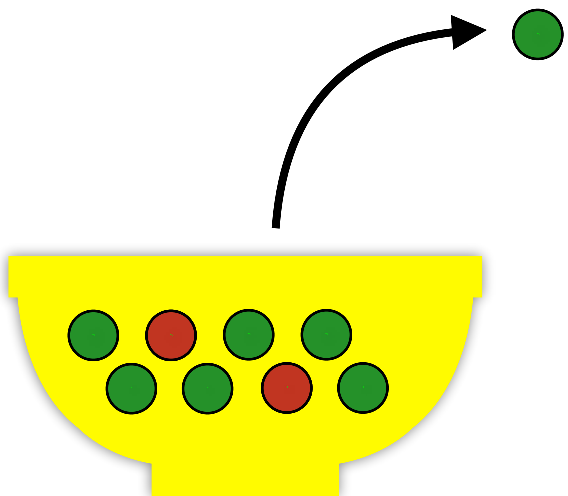
Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn $1 but an arbitrary number of other balls in urn become useless

** Although principle of deferred decision means this doesn't really matter.*

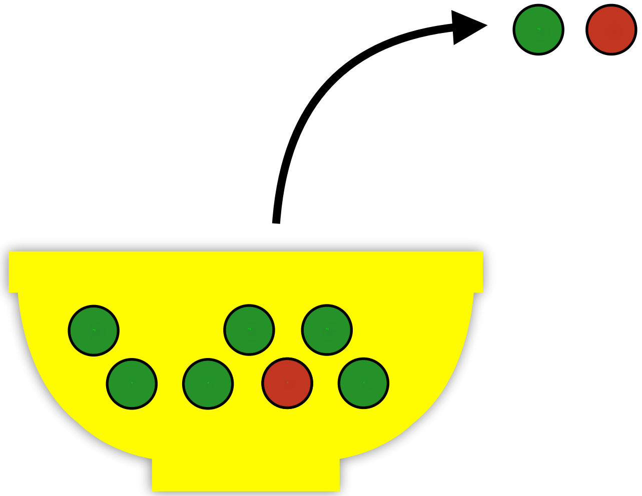
Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn $1 but an arbitrary number of other balls in urn become useless

** Although principle of deferred decision means this doesn't really matter.*

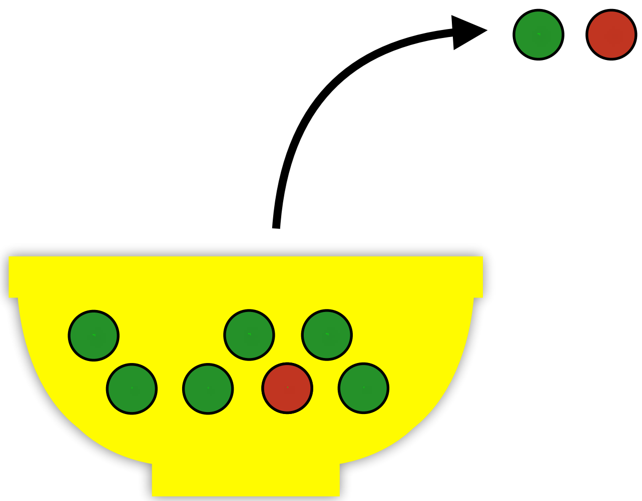
Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn \$1 but an arbitrary number of other balls in urn become useless

* *Although principle of deferred decision means this doesn't really matter.*

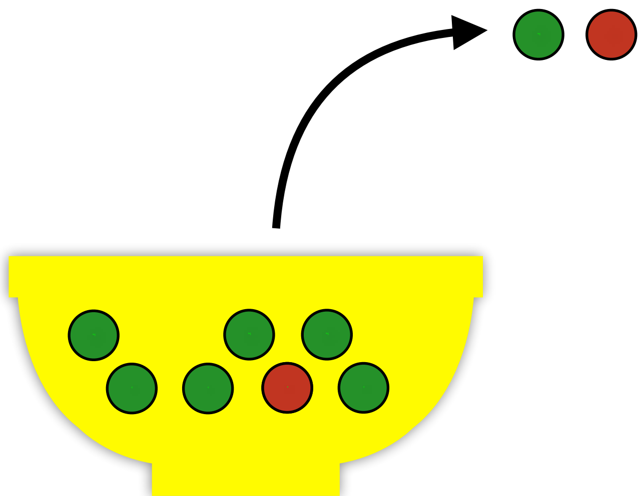
Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn $1 but an arbitrary number of other balls in urn become useless
- If fraction of useful balls stays above 1/2, we expect ≥ $k/2.

*Although principle of deferred decision means this doesn't really matter.

Proof of Lemma: Urn Analysis



- Consider urn with balls corresponding to useful sets
- Draw k balls one-at-a-time*. If we draw a useful ball we earn \$1 but an arbitrary number of other balls in urn become useless
- If fraction of useful balls stays above 1/2, we expect ≥ \$k/2.
- At end of each pass we remove all useless balls and restart.

* *Although principle of deferred decision means this doesn't really matter.*

Improving Space Complexity

Improving Space Complexity

- ***Definition*** Set is v-useful if it covers between v and 2v new elts.

Improving Space Complexity

- *Definition* Set is v-useful if it covers between v and 2v new elts.

- **Initialize:** Let $C=\emptyset$ be covered elements.
- **In parallel for $i \in \{1, \ldots, \log(2k)\}$**
  - Sample 2^i sets amongst the $OPT/2^i$ useful sets
  - Add set any sampled sets that remain useful
- **Repeat until we have covered $OPT/2$ elts**

Improving Space Complexity

- *Definition* Set is v-useful if it covers between v and 2v new elts.

  - **Initialize:** Let $C=\emptyset$ be covered elements.
  - **In parallel for $i \in \{1, \ldots, \log(2k)\}$**
    - Sample 2^i sets amongst the $OPT/2^i$ useful sets
    - Add set any sampled sets that remain useful
  - **Repeat until we have covered OPT/2 elts**

- *1/2 Approx* Carries over from insert-only analysis.

Improving Space Complexity

- *Definition* Set is v-useful if it covers between v and 2v new elts.

> - **Initialize:** Let $C=\emptyset$ be covered elements.
> - **In parallel for $i \in \{1, \ldots, \log(2k)\}$**
>   - Sample 2^i sets amongst the $OPT/2^i$ useful sets
>   - Add set any sampled sets that remain useful
> - **Repeat until we have covered OPT/2 elts**

- *1/2 Approx* Carries over from insert-only analysis.
- *Space* Can sample 2^i sets in $\approx 2^i \max_{i:OPT/2^i \text{ useful}} |S_i \setminus C| \leq \tilde{O}_\varepsilon(k)$ space

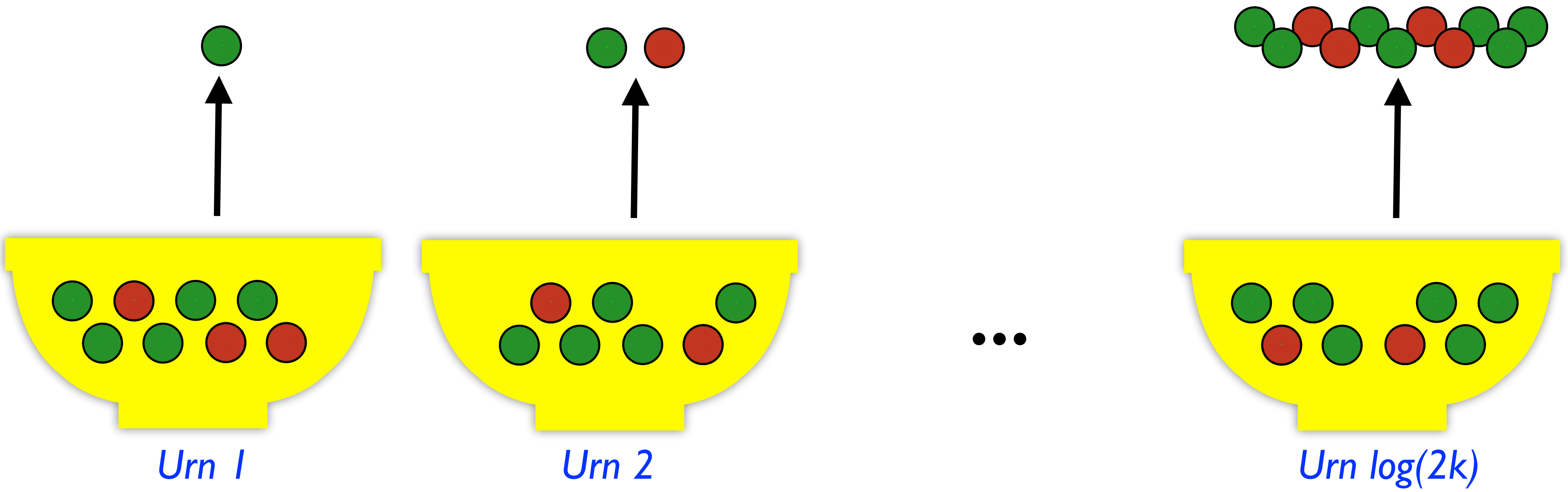
Improving Space Complexity

- *Definition* Set is v-useful if it covers between v and 2v new elts.

- **Initialize:** Let $C=\emptyset$ be covered elements.
- **In parallel for $i \in \{1, \ldots, \log(2k)\}$**
  - Sample 2^i sets amongst the $OPT/2^i$ useful sets
  - Add set any sampled sets that remain useful
- **Repeat until we have covered OPT/2 elts**

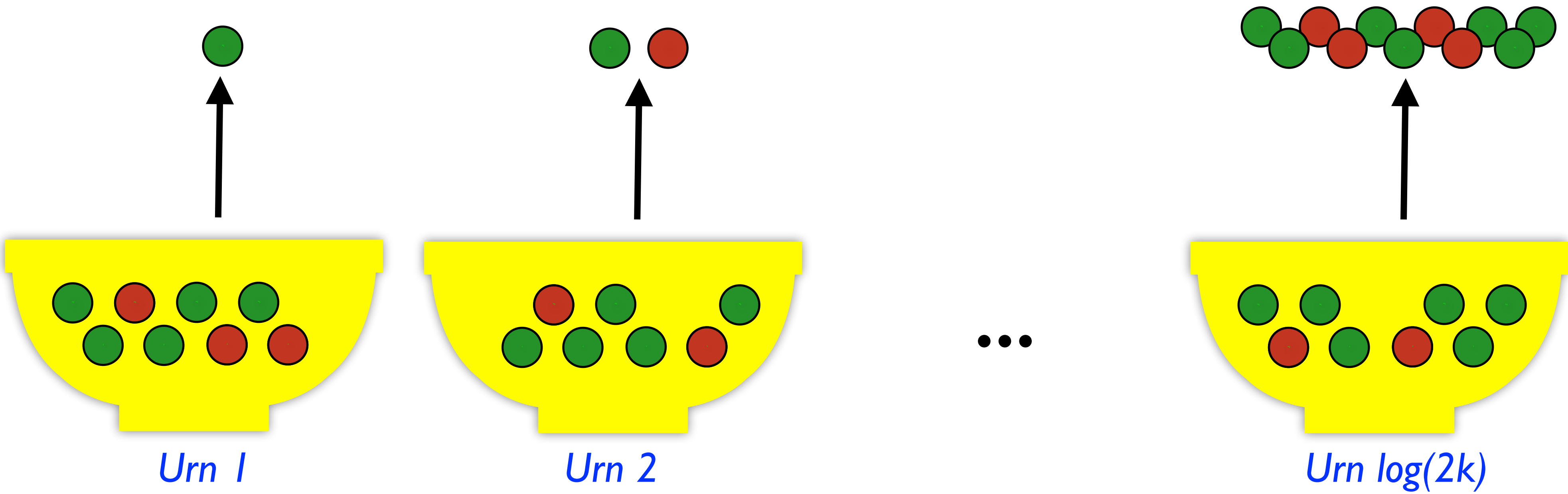
- *1/2 Approx* Carries over from insert-only analysis.
- *Space* Can sample 2^i sets in $\approx 2^i \max_{i:OPT/2^i \text{ useful}} |S_i \setminus C| \leq \tilde{O}_\varepsilon(k)$ space
- *Lemma* O(log k + log m/log log m) passes.

Proof Idea of Lemma: Cascading Urns



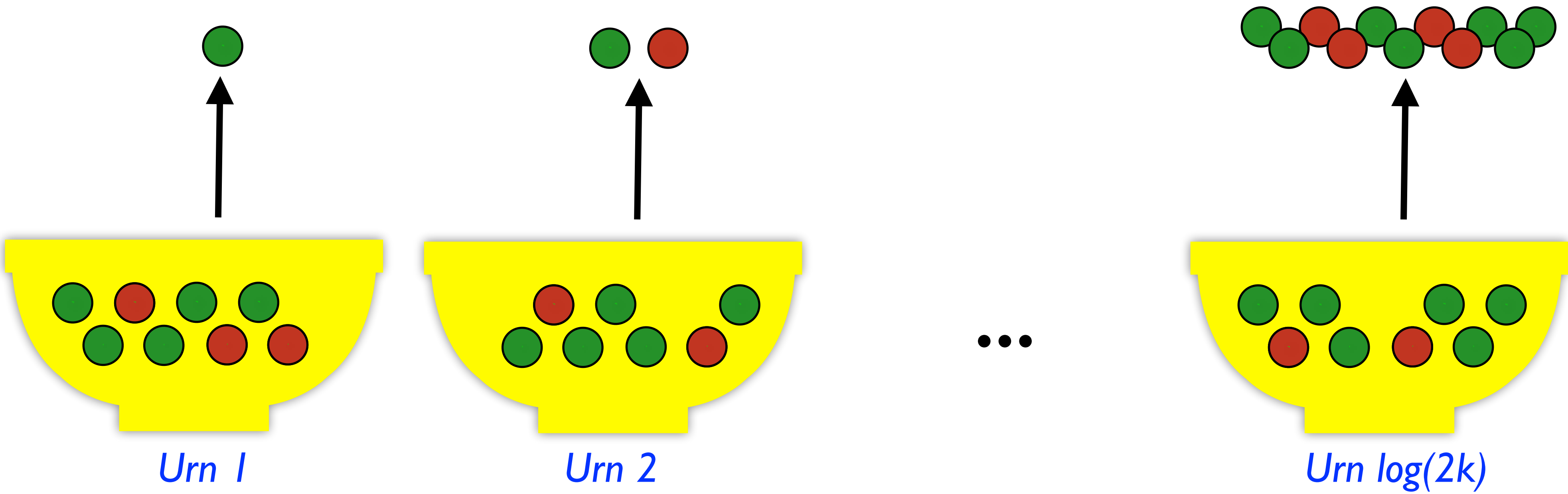
- Balls in i-th urn correspond to ($OPT/2^i$)-useful sets.

Proof Idea of Lemma: Cascading Urns



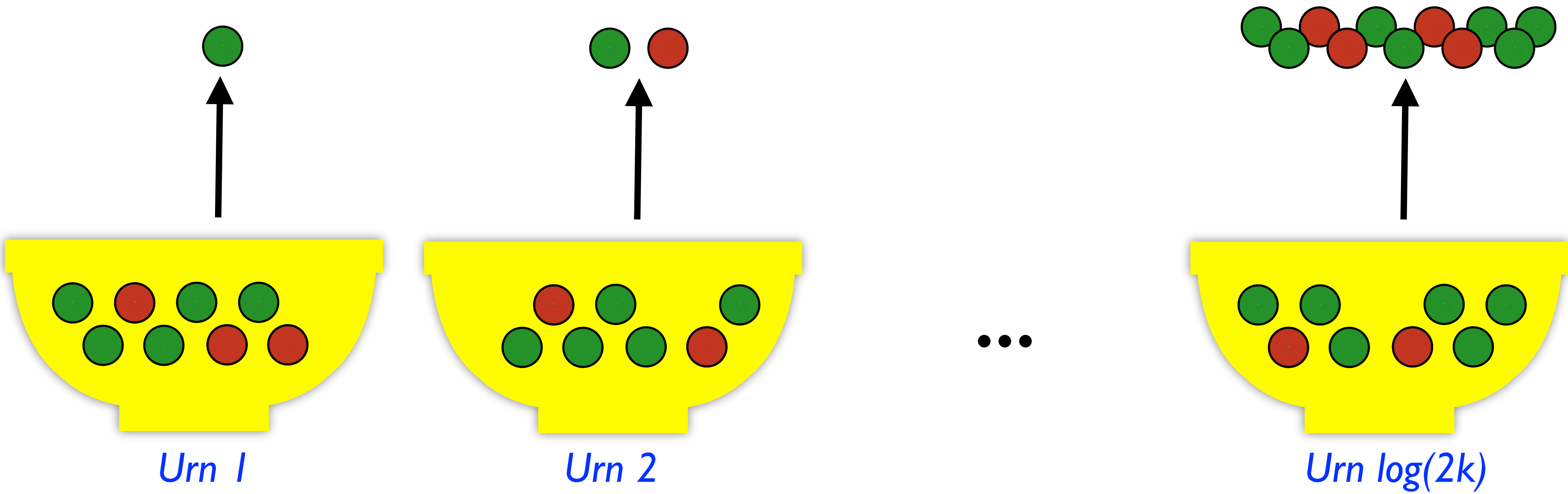
- Balls in i-th urn correspond to ($OPT/2^i$)-useful sets.
- Draw 2^i balls from Urn i. May cause other balls to become useless.

Proof Idea of Lemma: Cascading Urns



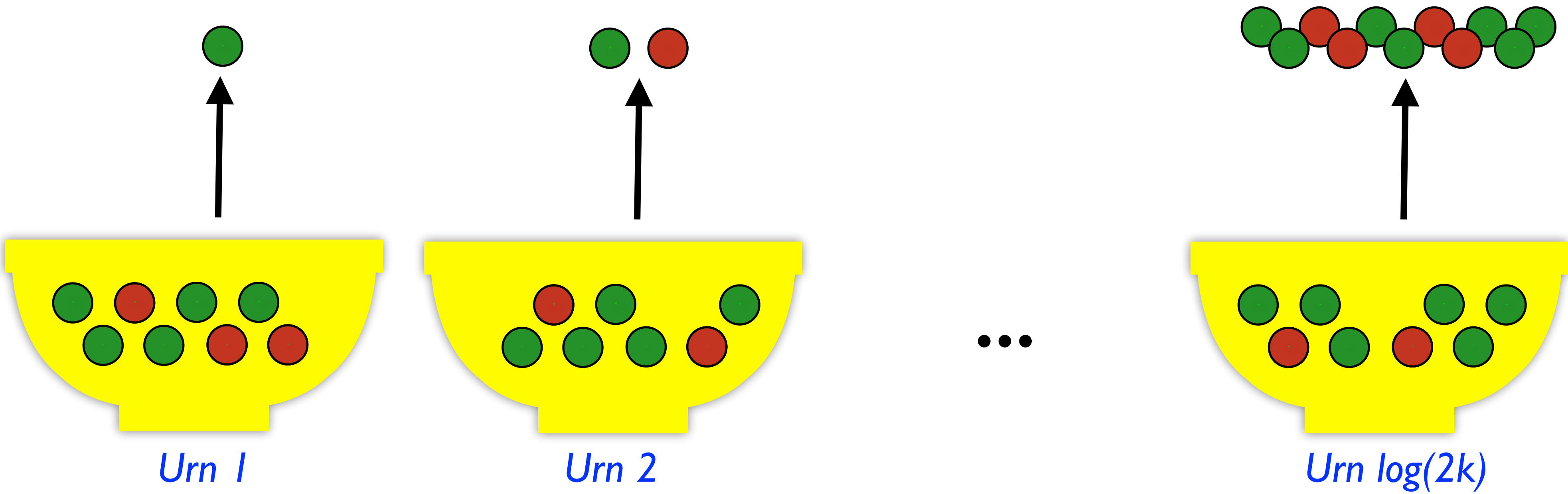
- Balls in i-th urn correspond to $(OPT/2^i)$-useful sets.
- Draw 2^i balls from Urn i. May cause other balls to become useless.
- A useful ball from Urn i gives \$$OPT/2^i$

Proof Idea of Lemma: Cascading Urns



- Balls in i-th urn correspond to $(OPT/2^i)$-useful sets.
- Draw 2^i balls from Urn i. May cause other balls to become useless.
- A useful ball from Urn i gives $\$OPT/2^i$
- If fraction of useful balls stays above 1/2, we expect $\geq \$OPT/2$.

Proof Idea of Lemma: Cascading Urns



- Balls in i-th urn correspond to (OPT/2^i)-useful sets.
- Draw 2^i balls from Urn i. May cause other balls to become useless.
- A useful ball from Urn i gives \$OPT/2^i
- If fraction of useful balls stays above 1/2, we expect ≥ \$OPT/2.
- At end of each pass useless balls are removed and potentially placed in later urns, i.e., balls "cascade".

Getting 1-1/e-ε Approx

Getting $1-1/e-\varepsilon$ Approx

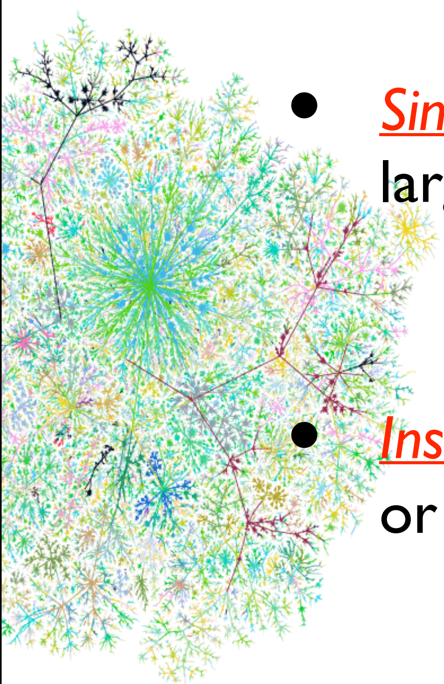
- **Initialize:** Let $C=\emptyset$ be covered elements.
- **In parallel for $i\in\{1, \ldots, \log k\}$**
  - Sample 2^i sets amongst the $OPT/2^i$ useful sets
  - Add set any sampled sets that remain useful
- **In series for $i\in\{1, \ldots, \log_{1+\varepsilon} e\}$**
  - Sample k sets amongst $(1+\varepsilon)^{-i}\, OPT/k$ useful sets
  - Add set any sampled sets that remain useful

Getting 1-1/e-ε Approx

- **Initialize:** Let $C=\emptyset$ be covered elements.
- **In parallel for $i\in\{1, \ldots, \log k\}$**
  - Sample 2^i sets amongst the $OPT/2^i$ useful sets
  - Add set any sampled sets that remain useful
- **In series for $i\in\{1, \ldots, \log_{1+\varepsilon} e\}$**
  - Sample k sets amongst $(1+\varepsilon)^{-i}\, OPT/k$ useful sets
  - Add set any sampled sets that remain useful

- *Theorem* $O(\log m + \varepsilon^{-1} \log m/\log\log m)$ pass, $\tilde{O}(\varepsilon^{-2}k)$-space, $1-1/e-\varepsilon$ approx. in the dynamic set stream model,

Summary of Talk



- *Simplification* If sets are small, can throw out many sets. If sets are large, can subsample universe to ensure sets are small-ish.

  Older results: [*Demaine et al.* DISC 14], [*McGregor, Vu.* Theory Comput. Syst. 19], [*McGregor et al.* ICDT 21]

- *Insert-Only Streams* $\tilde{O}_\varepsilon(k)$ space suffices for 1/2-ε approx (1-pass) or 1-1/e-ε (1-pass random order $O(1/\varepsilon)$-pass arbitrary order).

  Prev. best space bound for random order $\tilde{O}_\varepsilon(k^2)$ [*Warneke et al.* ESA 23]

- *Dynamic Streams* $\tilde{O}(\varepsilon^{-2} k)$ space, $O(\log m+\varepsilon^{-1} \log m/\log \log m)$ passes suffice for 1-1/e-ε approx.

  [*Chakrabarti, McGregor, Wirth.* ESA 24]
  Prev. best space bound $\tilde{O}(n+\varepsilon^{-4} k)$ [*Assadi, Khanna.* SODA 18]

Thanks!