Set Cover in Sub-linear Time

Piotr Indyk
MIT

Sepideh Mahabadi
Columbia University

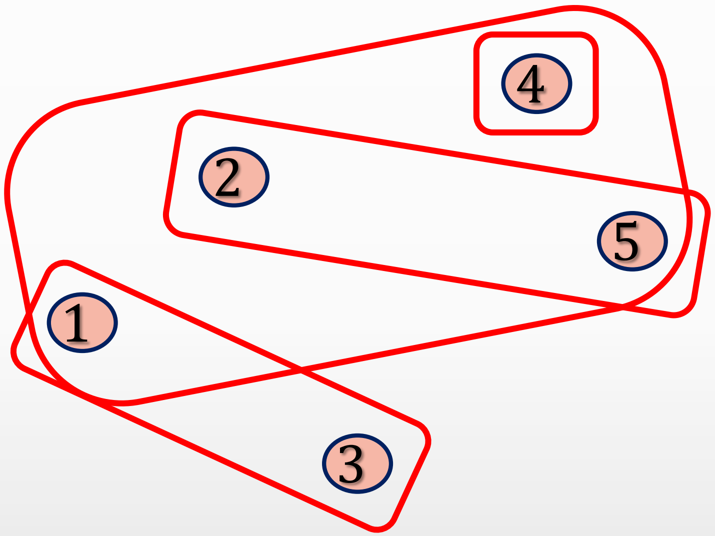
Ronitt Rubinfeld
MIT/TAU

Ali Vakilian
MIT

Anak Yodpinyanee
MIT

Set Cover Problem

Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$

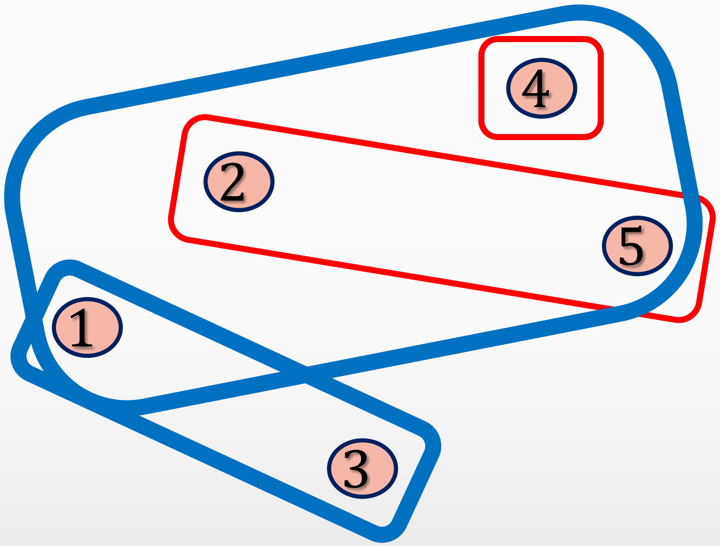


Set Cover Problem

Input: Collection $\mathcal{F}$ of sets $S_1, \dots, S_m$, each a subset of $\mathcal{U} = \{1, \dots, n\}$

Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:

- $\mathcal{C}$ covers $\mathcal{U}$
- $|\mathcal{C}|$ is minimized



Set Cover Problem

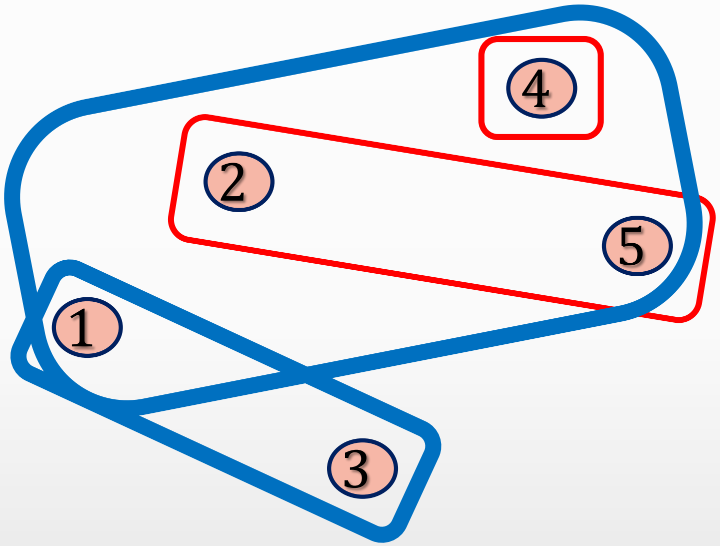
Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$

Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:

- $\mathcal{C}$ covers $\mathcal{U}$
- $|\mathcal{C}|$ is minimized

Complexity:

- NP-hard

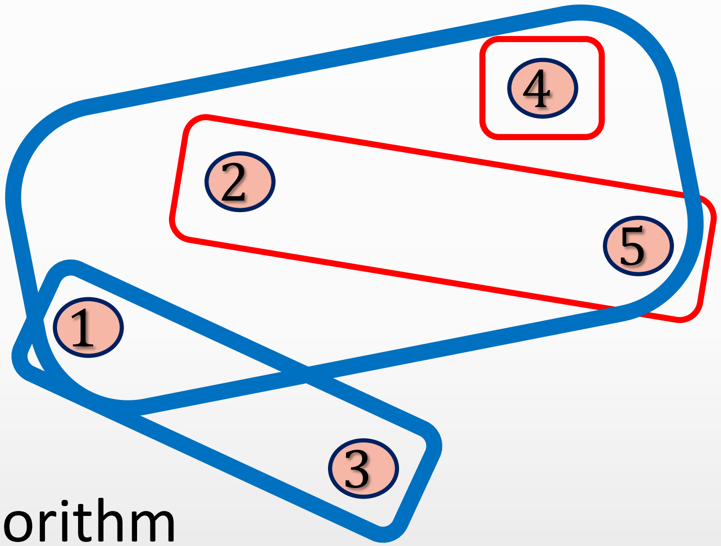


Set Cover Problem

Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$

Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:

- $\mathcal{C}$ covers $\mathcal{U}$
- $|\mathcal{C}|$ is minimized



Complexity:

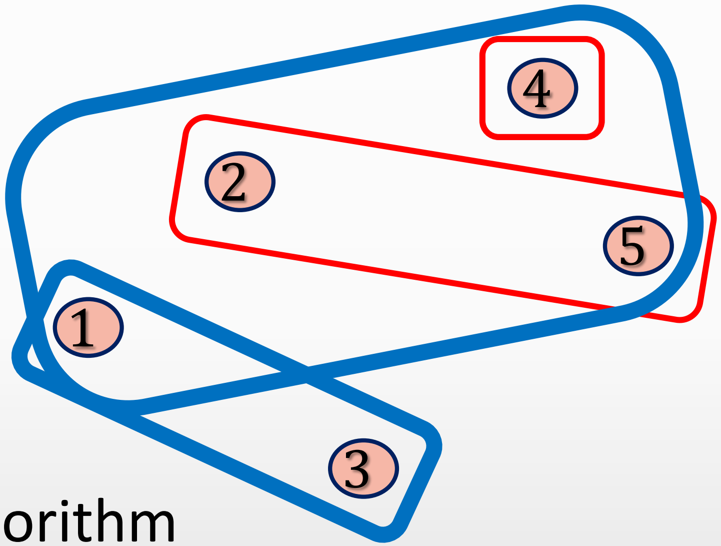
- NP-hard
- Greedy $(\ln n)$-approximation algorithm

Set Cover Problem

Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$

Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:

- $\mathcal{C}$ covers $\mathcal{U}$
- $|\mathcal{C}|$ is minimized



Complexity:

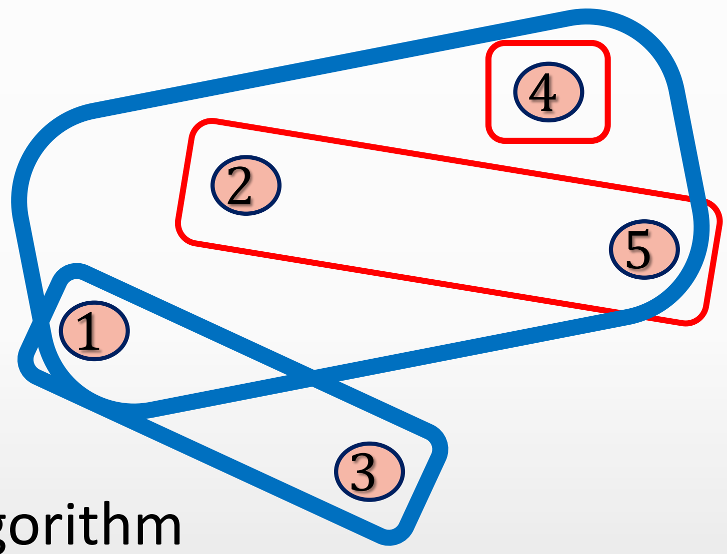
- NP-hard
- Greedy $(\ln n)$-approximation algorithm
- Can't do better unless **P=NP** [LY91][RS97][Fei98][AMS06][DS14]

Set Cover Problem

Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$

Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:

- $\mathcal{C}$ covers $\mathcal{U}$
- $|\mathcal{C}|$ is minimized



Complexity:

- NP-hard
- Greedy $(\ln n)$-approximation algorithm
- Can't do better unless **P=NP** [LY91][RS97][Fei98][AMS06][DS14]

*"Is it possible to solve minimum set cover in **sub-linear time**?"*

Sub-linear Time Set Cover

Data Access Model ?

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

EltOf(S, i): ith element in $\boldsymbol{S}$
SetOf(e, j): jth set containing $\boldsymbol{e}$

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs

EltOf(S, i): ith element in S
SetOf(e, j): jth set containing e

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in $\boldsymbol{mn}$

EltOf$(\boldsymbol{S}, \boldsymbol{i})$: ith element in $\boldsymbol{S}$
SetOf$(\boldsymbol{e}, \boldsymbol{j})$: jth set containing $\boldsymbol{e}$

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

> **EltOf(S, i)**: ith element in S
> **SetOf(e, j)**: jth set containing e

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in mn

Prior Results

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

EltOf(S, i): ith element in S
SetOf(e, j): jth set containing e

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in mn

Prior Results

- [Nguyen, Onak'08][Yoshida, Yamamoto, Ito'12]
  - Constant queries, if degree is constant

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

EltOf(S, i): ith element in S
SetOf(e, j): jth set containing e

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in mn

Prior Results

- [Nguyen, Onak'08][Yoshida, Yamamoto, Ito'12]
  - Constant queries, if degree is constant
- [Koufogiannakis, Young'14][Grigoriadis, Kachiyan'95]:
  - Find $(1 + \epsilon)$-approximate **fractional solution**, then perform **randomized rounding** to achieve $O(\log n)$-approximation

Sub-linear Time Set Cover

Data Access Model [NO'08,YYI'12]

EltOf(S, i): ith element in S
SetOf(e, j): jth set containing e

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in mn

Prior Results

- [Nguyen, Onak'08][Yoshida, Yamamoto, Ito'12]
  - Constant queries, if degree is constant
- [Koufogiannakis, Young'14][Grigoriadis, Kachiyan'95]:
  - Find $(1+\epsilon)$-approximate **fractional solution**, then perform **randomized rounding** to achieve $O(\log n)$-approximation
  - $O(mk^2 + nk^2)$ (can be improved to $O(m + nk)$)

n = number of *elements* m = number of *sets* k = size of the optimal solution

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m \left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m \left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

ρ = approximation factor for offline **Set Cover**

n = number of *elements* m = number of *sets* k = Size of the optimal Solution

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

$\boldsymbol{\rho}$ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\widetilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\widetilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\widetilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* m = number of *sets* k = Size of the optimal Solution

Part one: upper bound

Theorem: There exists an algorithm that with high probability finds an $O(\rho\alpha)$-approximate cover which uses $\tilde{O}(\boldsymbol{mn^{1/\alpha}} + \boldsymbol{nk})$ number of queries.

Part one: upper bound

Theorem: There exists an algorithm that with high probability finds an $\mathrm{O}(\rho\alpha)$-approximate cover which uses $\tilde{O}(\boldsymbol{mn^{1/\alpha}} + \boldsymbol{nk})$ number of queries.

1. Two simple components used for coverage problems in massive data models.
   - Set Sampling
   - Element Sampling
2. The algorithm overview

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m}\log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

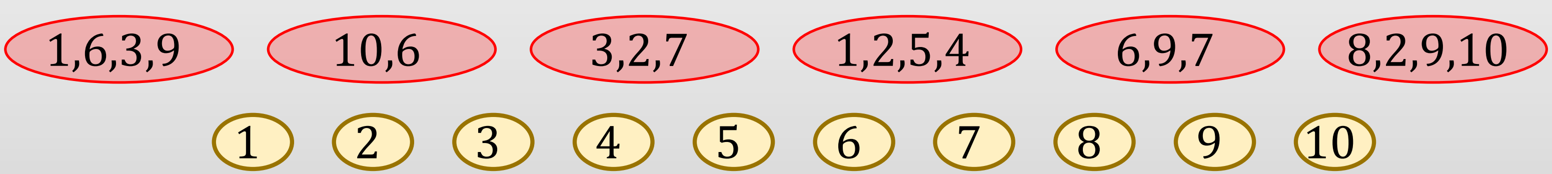
How we use the lemma: set $\ell = O(k)$

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

$\ell = 2$

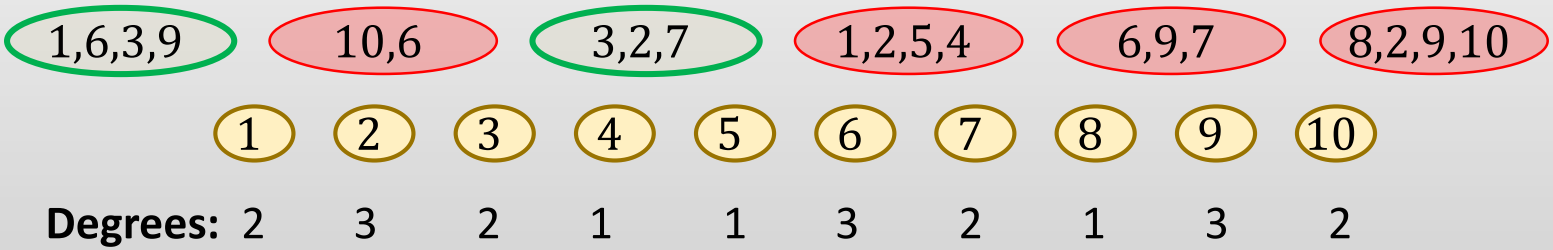


Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\text{m}\log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

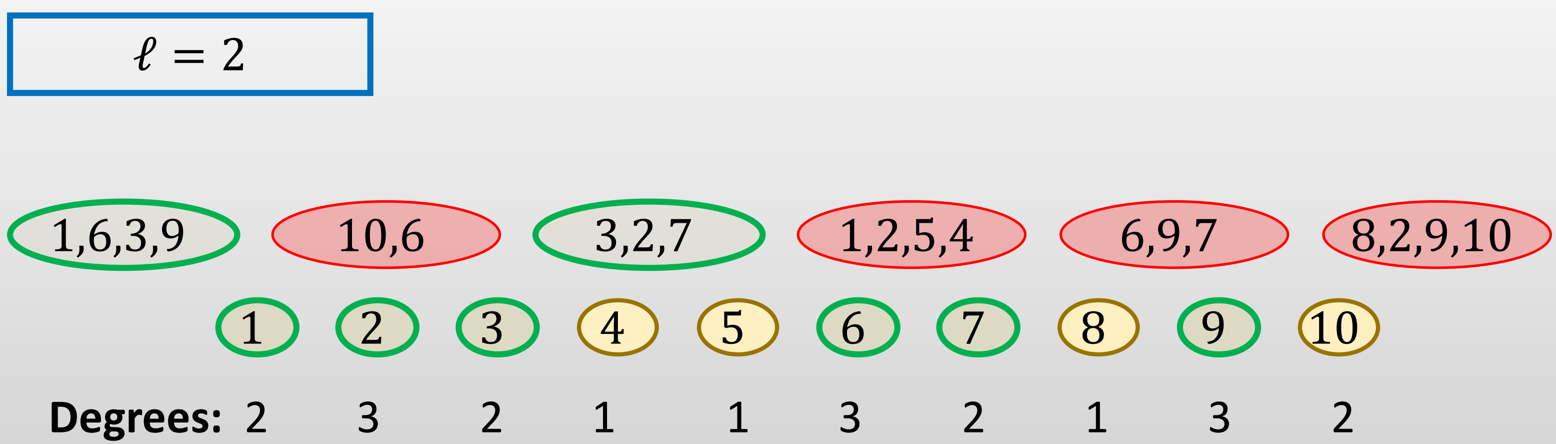
$\ell = 2$



Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m}\log n}{\ell}$ are covered w.h.p.

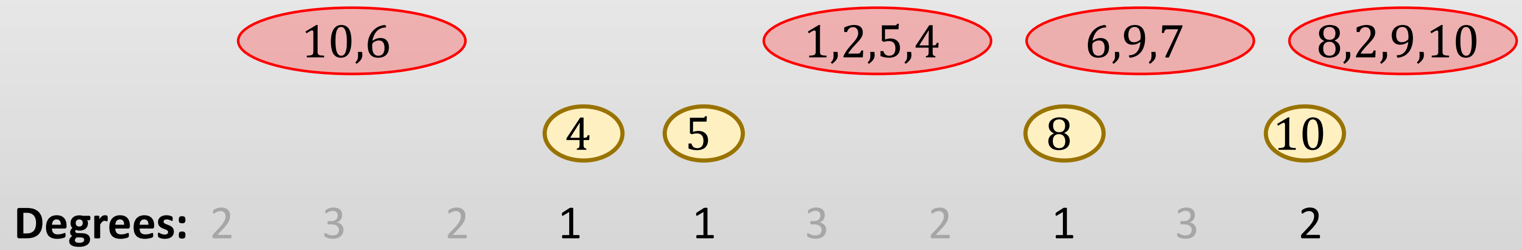
- We only need to worry about low degree elements.



Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

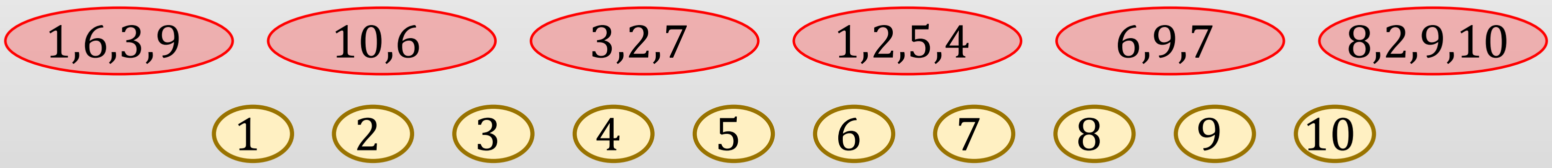


Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.

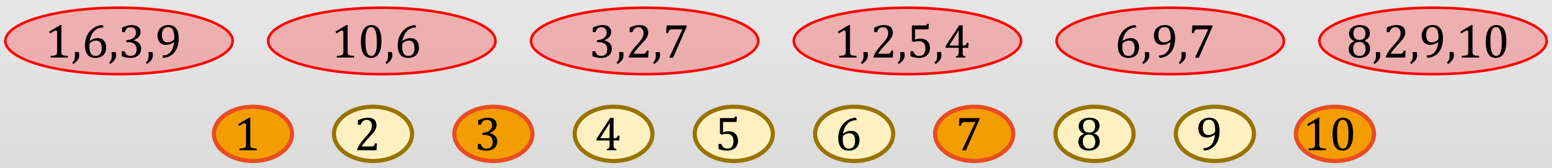
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



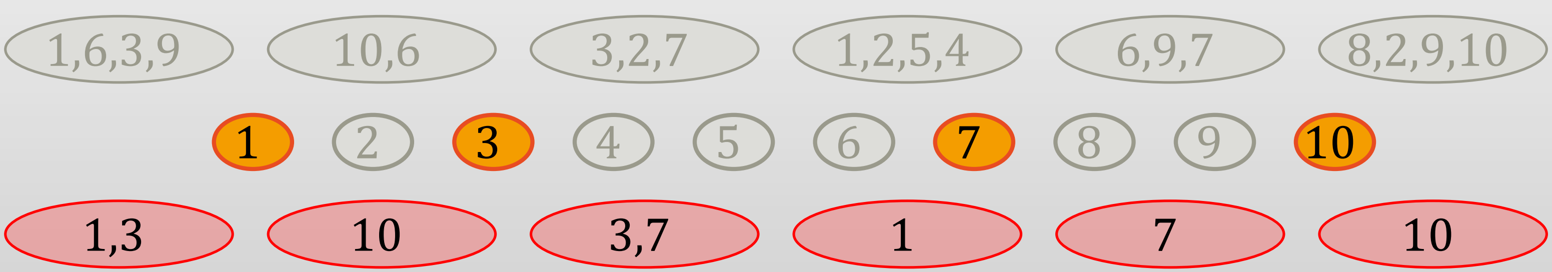
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



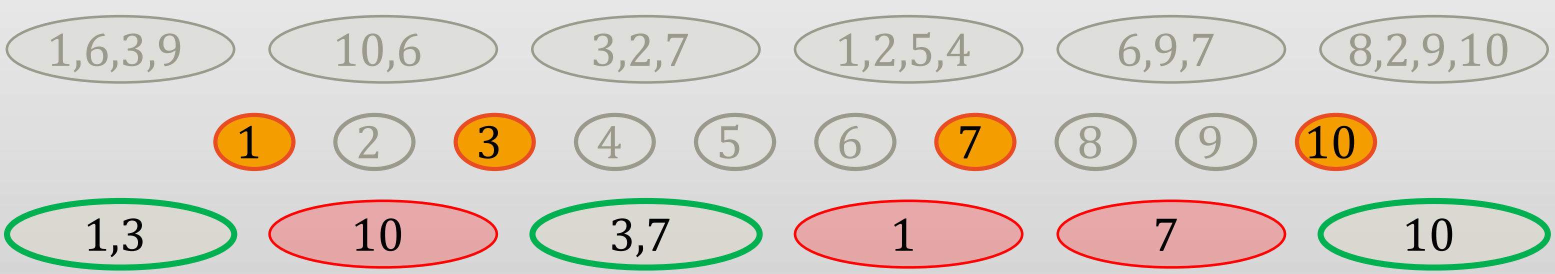
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



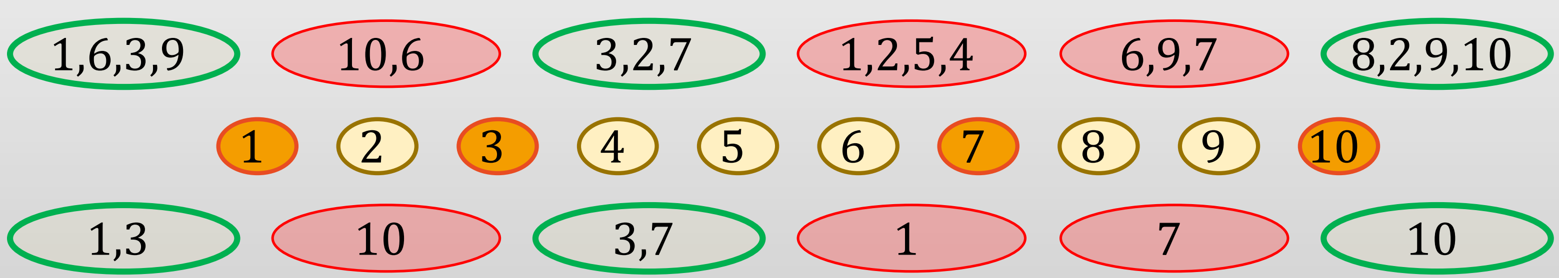
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



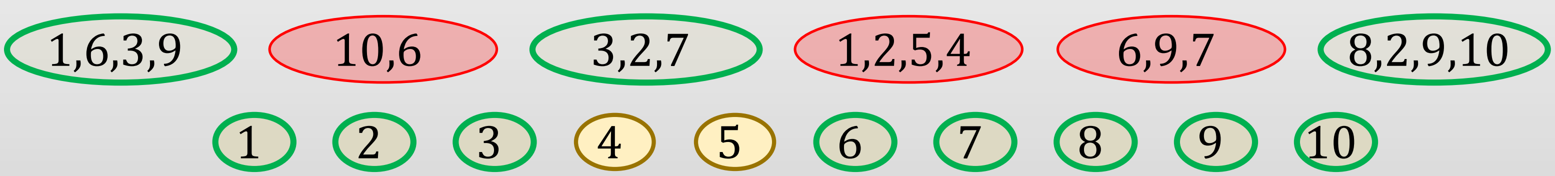
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



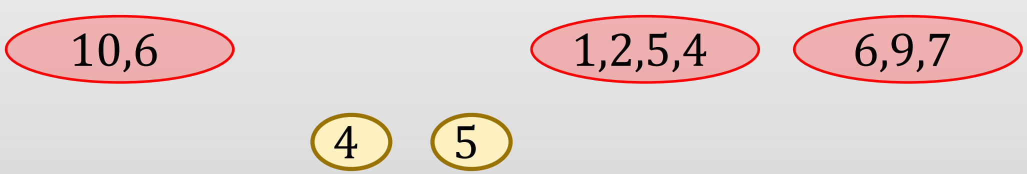
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



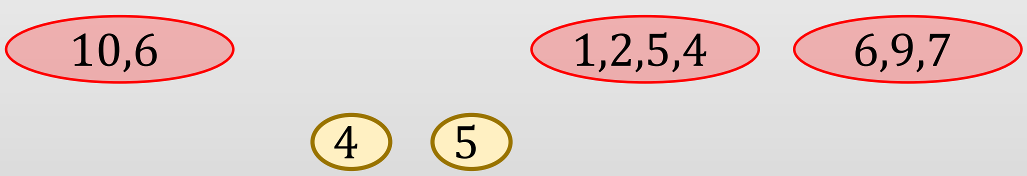
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



Component II: element sampling

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\delta})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\delta)$ fraction of the original elements w.h.p.



Algorithm

Make a guess ℓ of the value of the optimal solution k

Algorithm

Make a guess ℓ of the value of the optimal solution k

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

Algorithm

$\log n$ different guesses
$\ell \in \{1,2,4,\ldots,n\}$

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol $\leftarrow$ sampled sets

Algorithm

$\log n$ different guesses
$\ell \in \{1,2,4,\ldots,n\}$

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets

sample ℓ sets,
number of queries: $n\ell$

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m}\log n}{\ell}$ are covered w.h.p.

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol $\leftarrow$ sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1 - \frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

sample ℓ sets,
number of queries: $n\ell$

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1-\frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

sample ℓ sets,
number of queries: $n\ell$

$\boldsymbol{\delta = 1/n^{1/\alpha}}$

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\delta})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\delta)$ fraction of the original elements w.h.p.

Algorithm

$\log n$ different guesses
$\ell \in \{1,2,4,\dots,n\}$

Make a guess ℓ of the value of the optimal solution k

- Preprocessing: perform set sampling
- Sol $\leftarrow$ sampled sets
- For α iterations
  - Use element sampling to cover $(1-\frac{1}{n^{1/\alpha}})$ fraction of the uncovered elements.
  - Add the sets to Sol

sample ℓ sets,
number of queries: $n\ell$

sample $(\rho\ell n^{1/\alpha}\log m)$ elements,
number of queries:
$O\left(\rho\ell n^{1/\alpha}\log m\,\frac{m\log n}{\ell}\right)$
$=O(\rho m n^{1/\alpha}\log m\log n)$

$\boldsymbol{\delta = 1/n^{1/\alpha}}$

Element Sampling: Sampling $\Theta(\frac{\rho k\log m}{\delta})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\delta)$ fraction of the original elements w.h.p.

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1-\frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses $\ell \in \{1,2,4,\dots,n\}$

sample ℓ sets,
number of queries: $n\ell$

sample $(\rho\ell n^{1/\alpha}\log m)$ elements,
number of queries:
$O\left(\rho\ell n^{1/\alpha}\log m \frac{m\log n}{\ell}\right)$
$=O(\rho m n^{1/\alpha}\log m\log n)$

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol $\leftarrow$ sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1 - \frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

sample ℓ sets,
number of queries: $n\ell$

sample $(\rho\ell n^{1/\alpha} \log m)$ elements,
number of queries:
$O\left(\rho\ell n^{1/\alpha} \log m \frac{m \log n}{\ell}\right)$
$=O(\rho m n^{1/\alpha} \log m \log n)$

number of queries: $\rho n\ell$

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1-\frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.
- ❑ If all elements are covered, report Sol

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

sample ℓ sets,
number of queries: $n\ell$

sample $(\rho\ell n^{1/\alpha}\log m)$ elements,
number of queries:
$O\left(\rho\ell n^{1/\alpha}\log m\frac{m\log n}{\ell}\right)$
$=O(\rho m n^{1/\alpha}\log m\log n)$

number of queries: $\rho n\ell$

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For α iterations
  - Use element sampling to cover $(1 - \frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.
- ❑ If all elements are covered, report Sol

$\log n$ different guesses $\ell \in \{1,2,4,\ldots,n\}$

sample ℓ sets,
number of queries: $n\ell$

sample $(\rho\ell n^{1/\alpha} \log m)$ elements,
number of queries:
$O\left(\rho\ell n^{1/\alpha} \log m \frac{m \log n}{\ell}\right)$
$= O(\rho m n^{1/\alpha} \log m \log n)$

number of queries: $\rho n\ell$

Theorem: There exists an algorithm that with high probability finds an $O(\rho\alpha)$-approximate cover which uses $\tilde{O}(\boldsymbol{mn^{1/\alpha}} + \boldsymbol{nk})$ number of queries.

Results

Problem	Approximation	Constraints	Query Complexity
Set Cover	$\alpha\rho + 1$	$\alpha \geq 2$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$
	$\rho + 1$	$-$	$\tilde{O}\left(\frac{mn}{k}\right)$
	α	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$
	α	$\alpha \leq 1.01$ $k = O(n/\log m)$	$\tilde{\Omega}\left(\frac{mn}{k}\right)$
Cover Verification	$-$	$k \leq n/2$	$\tilde{\Omega}(nk)$

Cover Verification: given a set system, verify whether a given sub-collection of sets covers the universe.

ρ = approximation factor for offline **Set Cover**

n = number of *elements* **m = number of *sets*** **k = Size of the optimal Solution**

Part two: lower bound

Theorem: Any randomized algorithm that with probability at least 2/3 distinguishes whether the minimum Set Cover size is 2 or at least 3 requires $\tilde{\boldsymbol{\Omega}}(\boldsymbol{mn})$ number of queries.

High Level Approach

1. Construct a median instance I^*
   - Minimum Set Cover Size is 3

High Level Approach

1. Construct a median instance I^*
   - Minimum Set Cover Size is 3
2. Randomized Procedure on I^* to get a modified instance I
   - Minimum Set Cover Size is 2
   - I^* and I only differ in a few positions
   - The differences are distributed almost uniformly at random

High Level Approach

1. Construct a median instance I^*
   - Minimum Set Cover Size is 3
2. Randomized Procedure on I^* to get a modified instance I
   - Minimum Set Cover Size is 2
   - I^* and I only differ in a few positions
   - The differences are distributed almost uniformly at random

3. Any algorithm that can detect these two cases requires to query at least $\widetilde{\Omega}(mn)$ queries.

The Median Instance

Construction: is randomized. For every S, e the set S contains e with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

The Median Instance

Construction: is randomized. For every S, e the set S contains e with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

Properties: by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is $O(\log m)$
3. The intersection is at least $\Omega(n)$
4. For each element, the number of sets not covering it is at most $6m\sqrt{\frac{\log m}{n}}$
5. For any pair of elements the number of sets containing only the first element is at least $\frac{m\sqrt{9 \log m}}{4\sqrt{n}}$
6. For any three sets, the number of elements in the first two but not in the third one is at least $6\sqrt{n \log m}$

The Median Instance

Construction: is randomized. For every S, e the set S contains e with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

Properties: by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is $O(\log m)$
3. The intersection is at least $\Omega(n)$
4. For each element, the number of sets not covering it is at most $6m\sqrt{\frac{\log m}{n}}$
5. For any pair of elements the number of sets containing only the first element is at least $\frac{m\sqrt{9 \log m}}{4\sqrt{n}}$
6. For any three sets, the number of elements in the first two but not in the third one is at least $6\sqrt{n \log m}$

The Median Instance

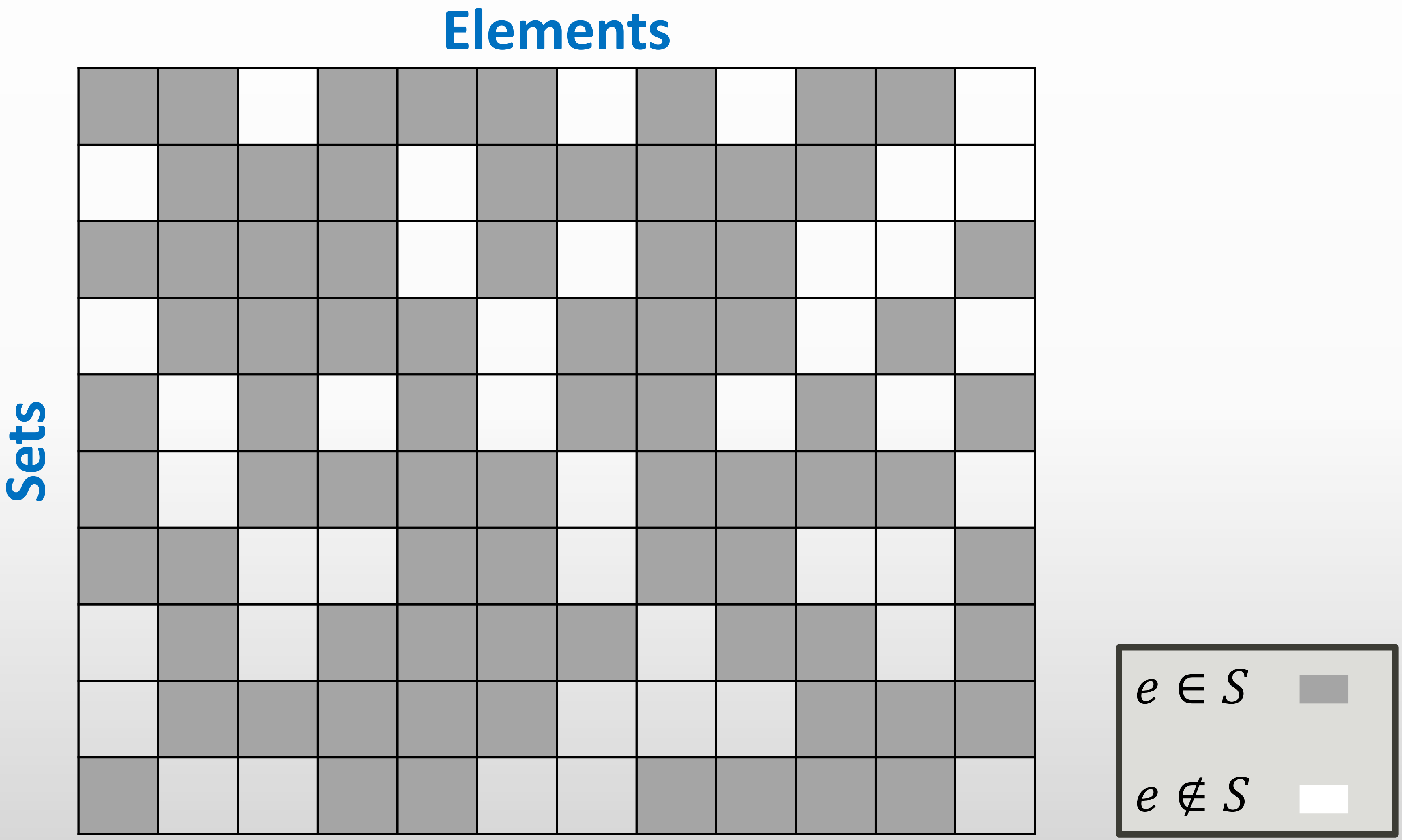
Construction: is randomized. For every S, e the set S contains e with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

Properties: by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is $O(\log m)$
3. The intersection is at least $\Omega(n)$
4. For each element, the number of sets not covering it is at most $6m\sqrt{\frac{\log m}{n}}$
5. For any pair of elements the number of sets containing only the first element is at least $\frac{m\sqrt{9 \log m}}{4\sqrt{n}}$
6. For any three sets, the number of elements in the first two but not in the third one is at least $6\sqrt{n \log m}$

Take one such instance I^* with the above properties

The Median Instance



Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

$$\boldsymbol{U} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4\}$$

$$\boldsymbol{S}_1 = \{\boldsymbol{e}_2, \boldsymbol{e}_3\}$$
$$\boldsymbol{S}_2 = \{\boldsymbol{e}_2, \boldsymbol{e}_4\}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_2, e_4\}} \leftarrow \boldsymbol{e_1}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$

$$\boldsymbol{S_2 = \{e_2, e_4\}} \leftarrow \boldsymbol{e_1}$$

$$\rightarrow \boldsymbol{e_2}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2
  - Pick a random set S_3 that contains e_1 but not e_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_2, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_1\}}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2
  - Pick a random set S_3 that contains e_1 but not e_2
  - S_2 and S_3 swap e_1 and e_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_2, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_1\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_1, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_2\}}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2
  - Pick a random set S_3 that contains e_1 but not e_2
  - S_2 and S_3 swap e_1 and e_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_2, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_1\}}$$

Swap

Modified instance

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_1, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_2\}}$$

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2
  - Pick a random set S_3 that contains e_1 but not e_2
  - S_2 and S_3 swap e_1 and e_2

$$U = \{e_1, e_2, e_3, e_4\}$$

$$S_1 = \{e_2, e_3\}$$
$$S_2 = \{e_2, e_4\}$$
$$S_3 = \{e_4, e_1\}$$

Swap

Modified instance

$$S_1 = \{e_2, e_3\}$$
$$S_2 = \{e_1, e_4\}$$
$$S_3 = \{e_4, e_2\}$$

Only four positions changes in the query access model.

Generating a Modified Instance

Pick two random sets S_1 and S_2 and turn them into a set cover.
How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add e_1 to S_2
  - Remove an element $e_2 \in S_2 \cap S_1$ from S_2
  - Pick a random set S_3 that contains e_1 but not e_2
  - S_2 and S_3 swap e_1 and e_2

$$\boldsymbol{U = \{e_1, e_2, e_3, e_4\}}$$

Two in **ElemOf** oracles
+
Two in **SetOf** oracles

Swap

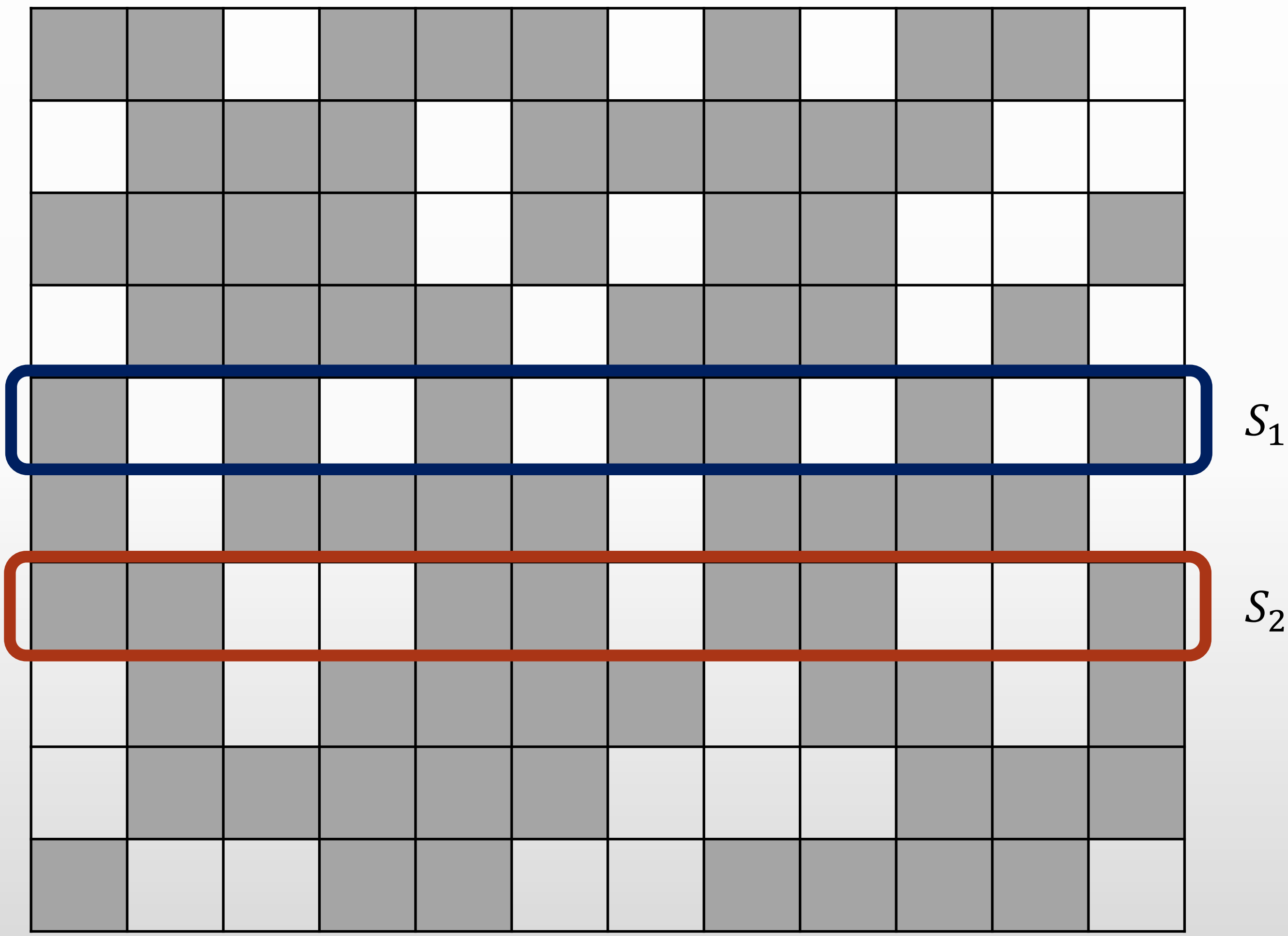
Modified instance

$$\boldsymbol{S_1 = \{e_2, e_3\}}$$
$$\boldsymbol{S_2 = \{e_1, e_4\}}$$
$$\boldsymbol{S_3 = \{e_4, e_2\}}$$

Only four positions changes in the query access model.

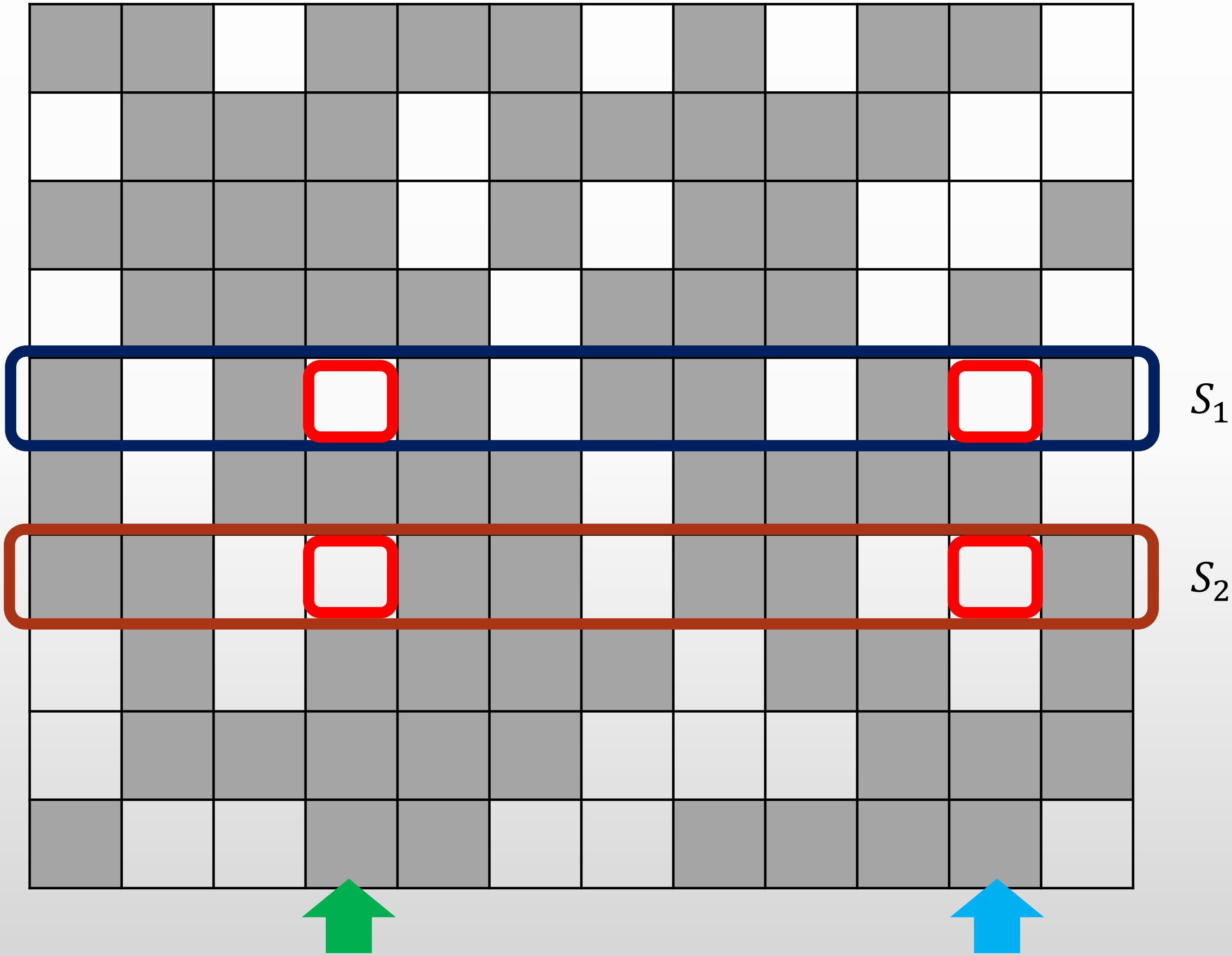
The Randomized Procedure

- Median Instance
- **Pick two Sets Uniformly at Random**



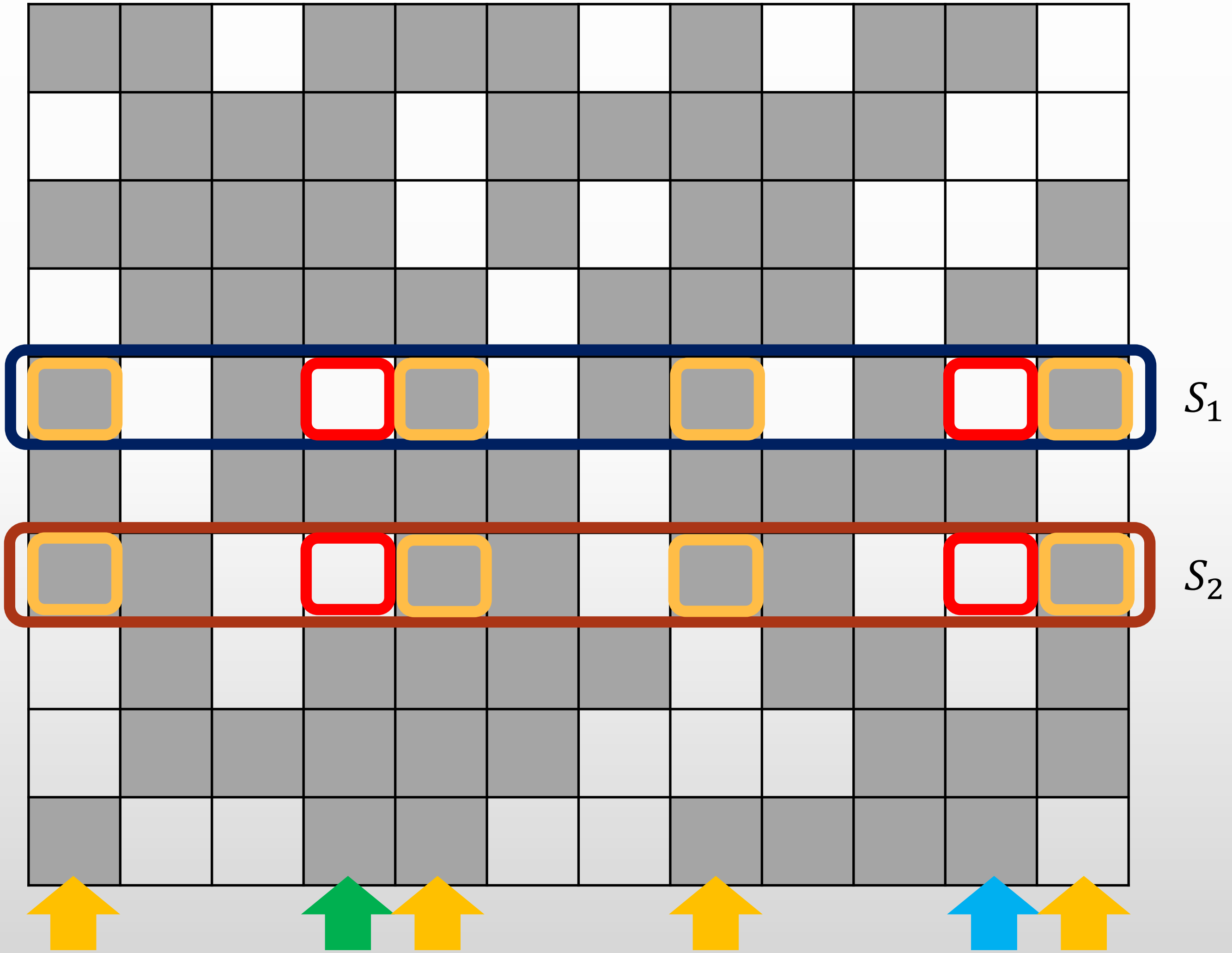
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- **Find the elements that are not covered**



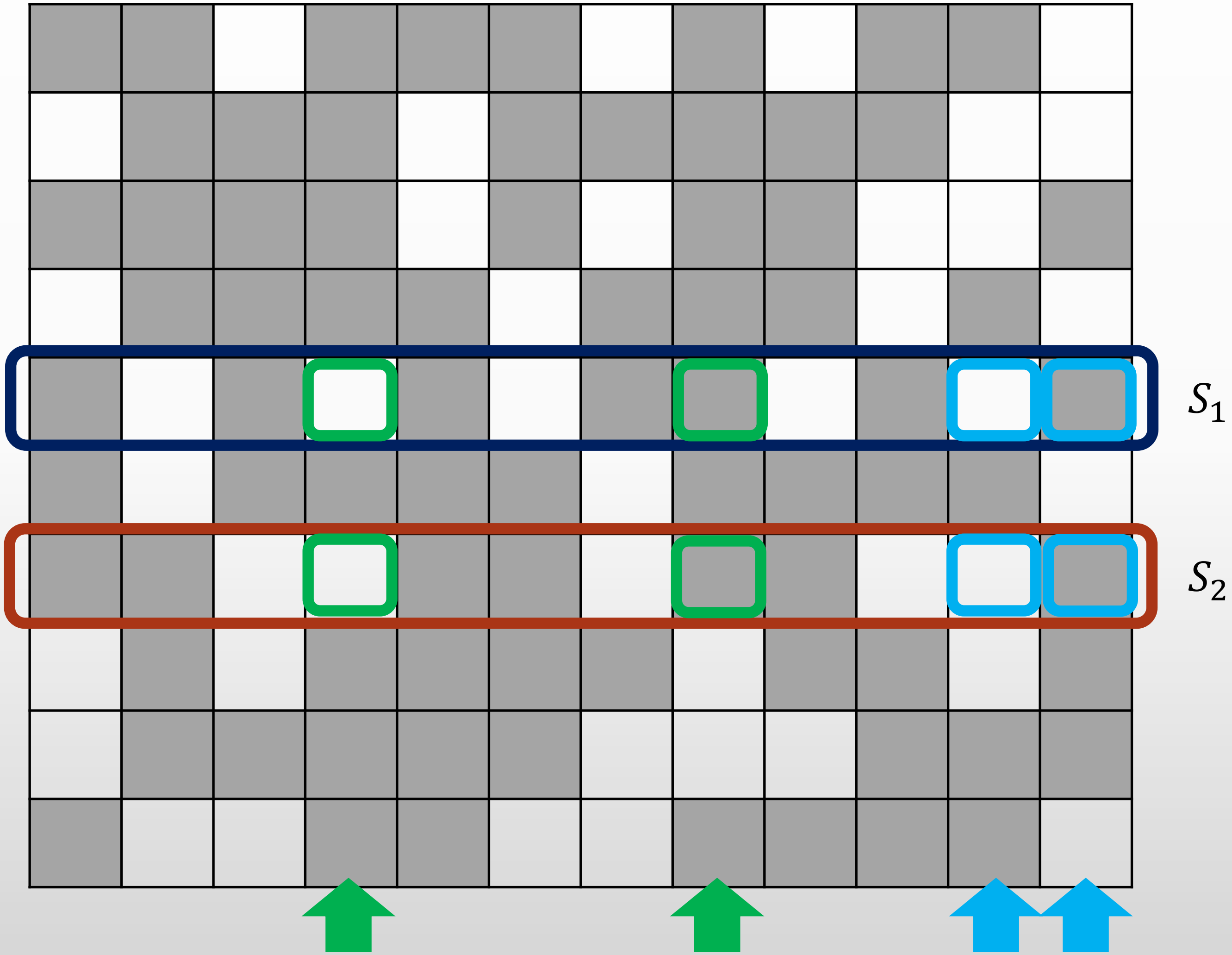
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- **Also find the elements that are covered by both**



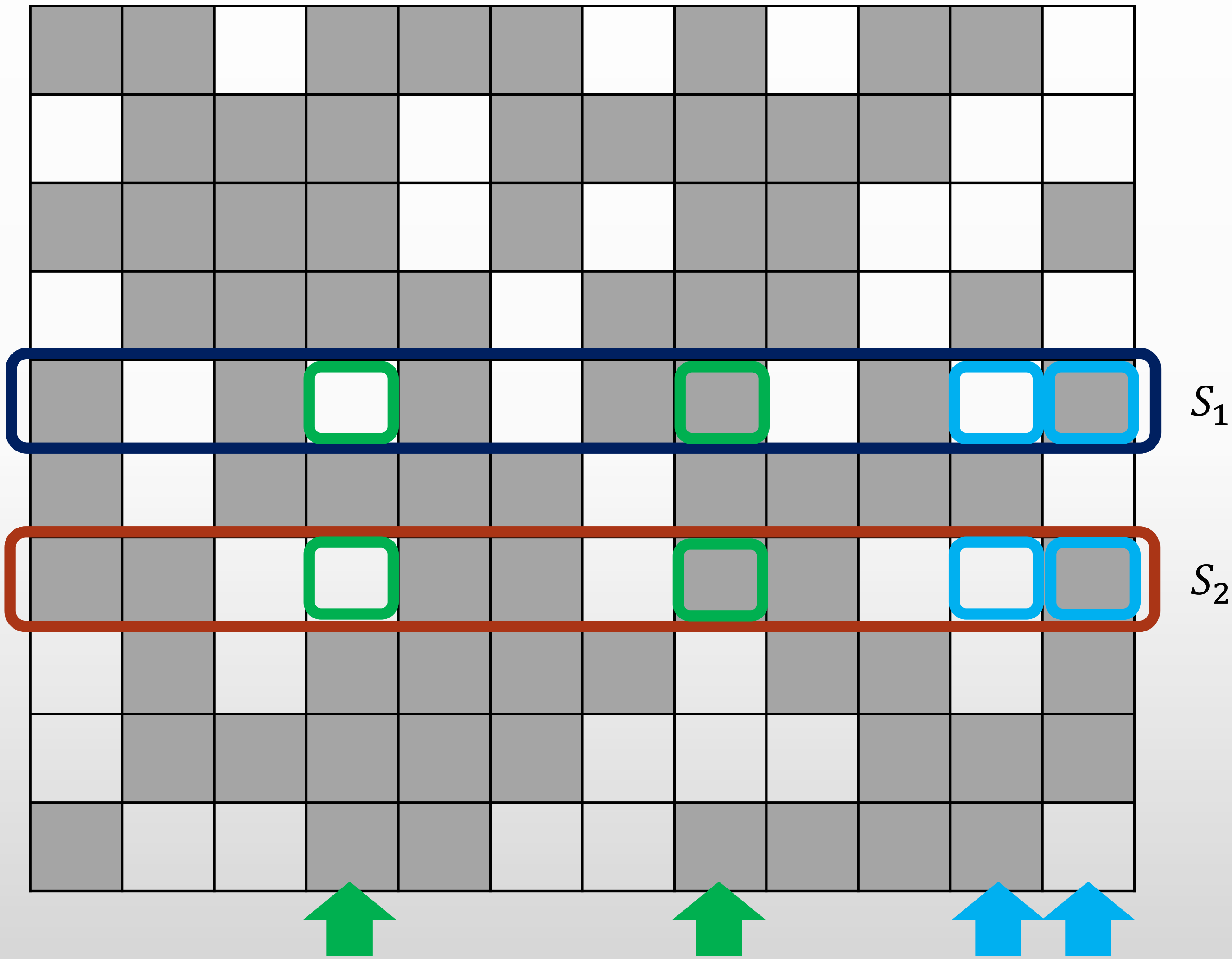
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- **Assign one element in the intersection to each uncovered element**



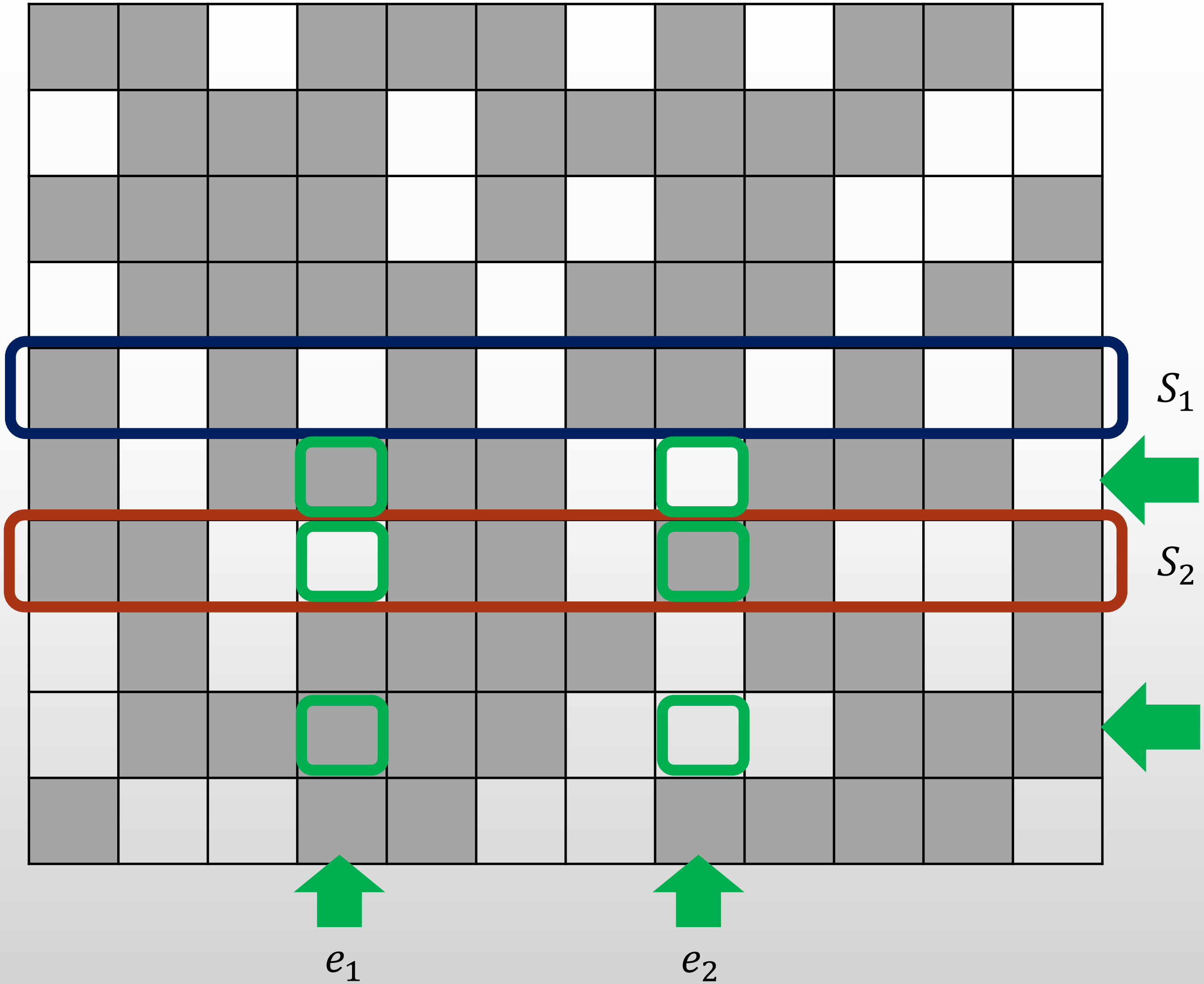
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- **Assign one element in the intersection to each uncovered element**



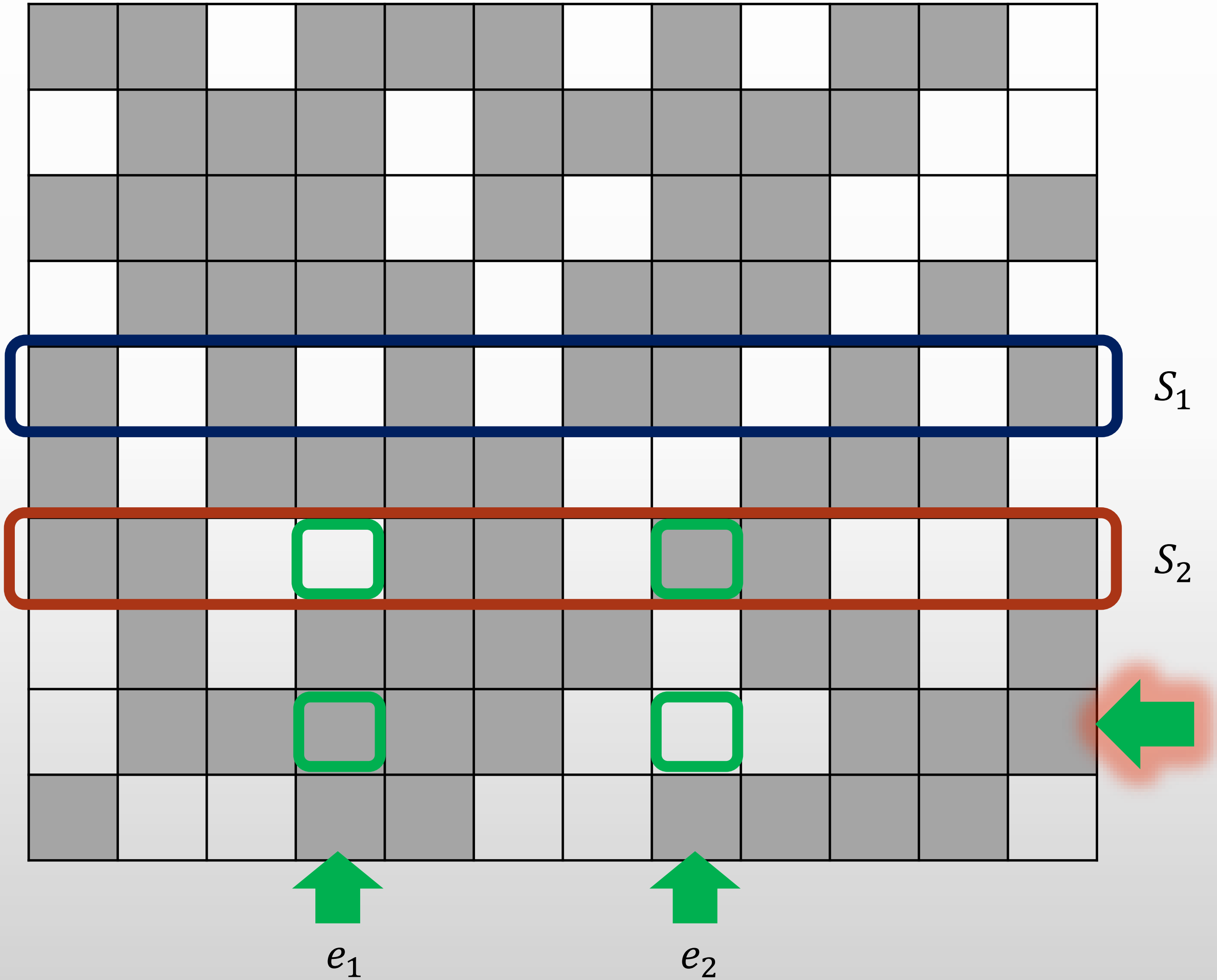
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- **Assign one element in the intersection to each uncovered element**



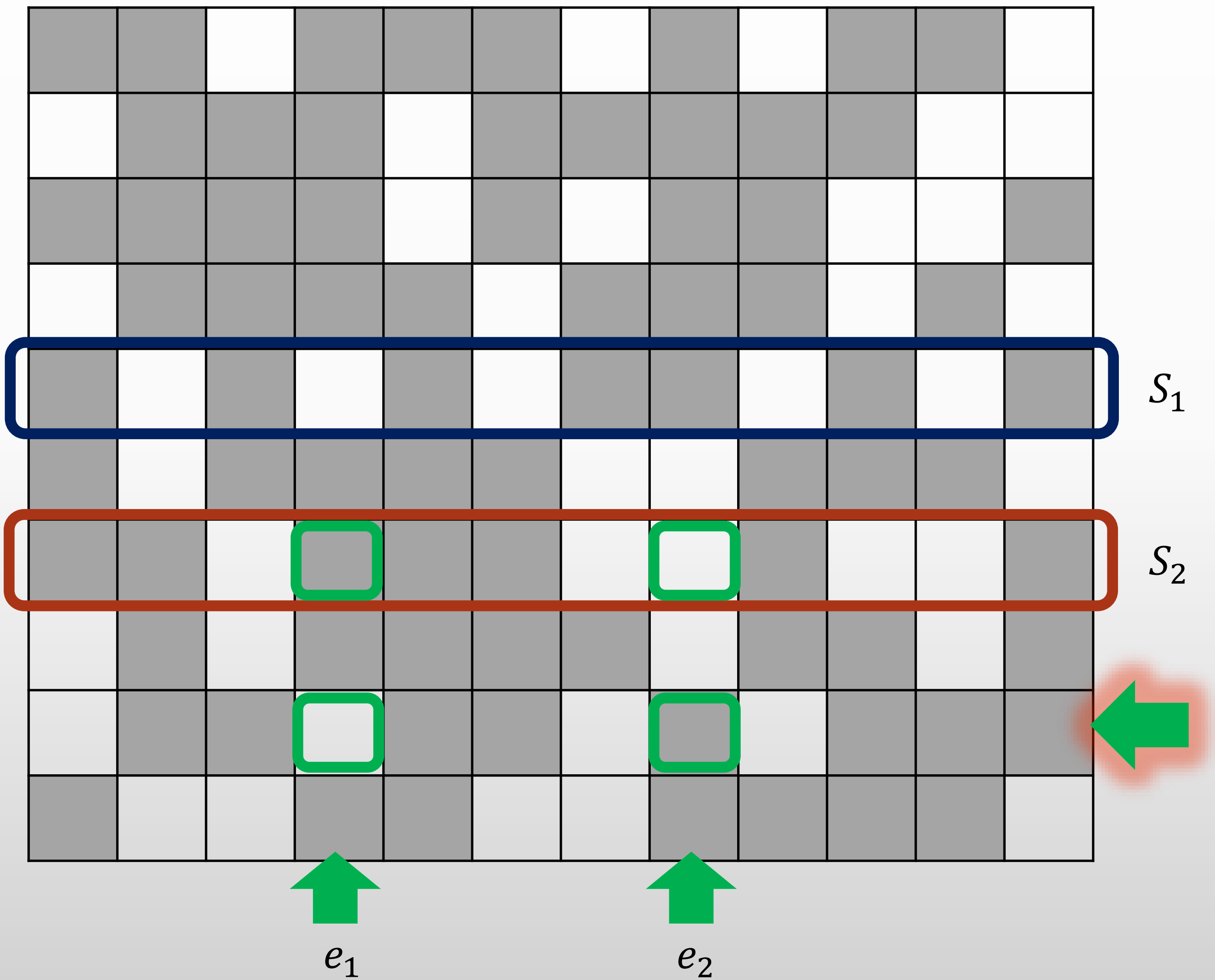
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
    - **Find a candidate set**



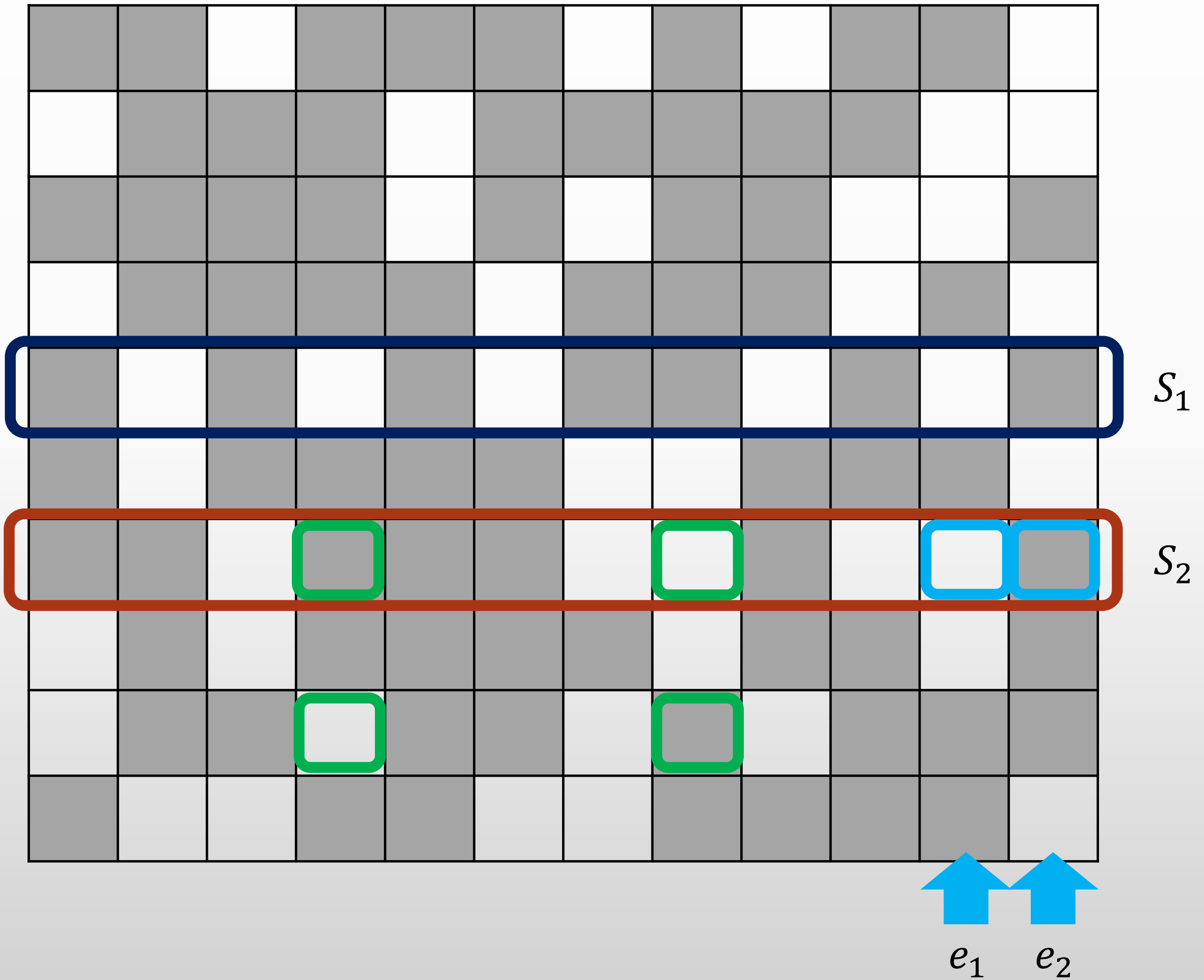
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
  - Find a candidate set
  - **swap**



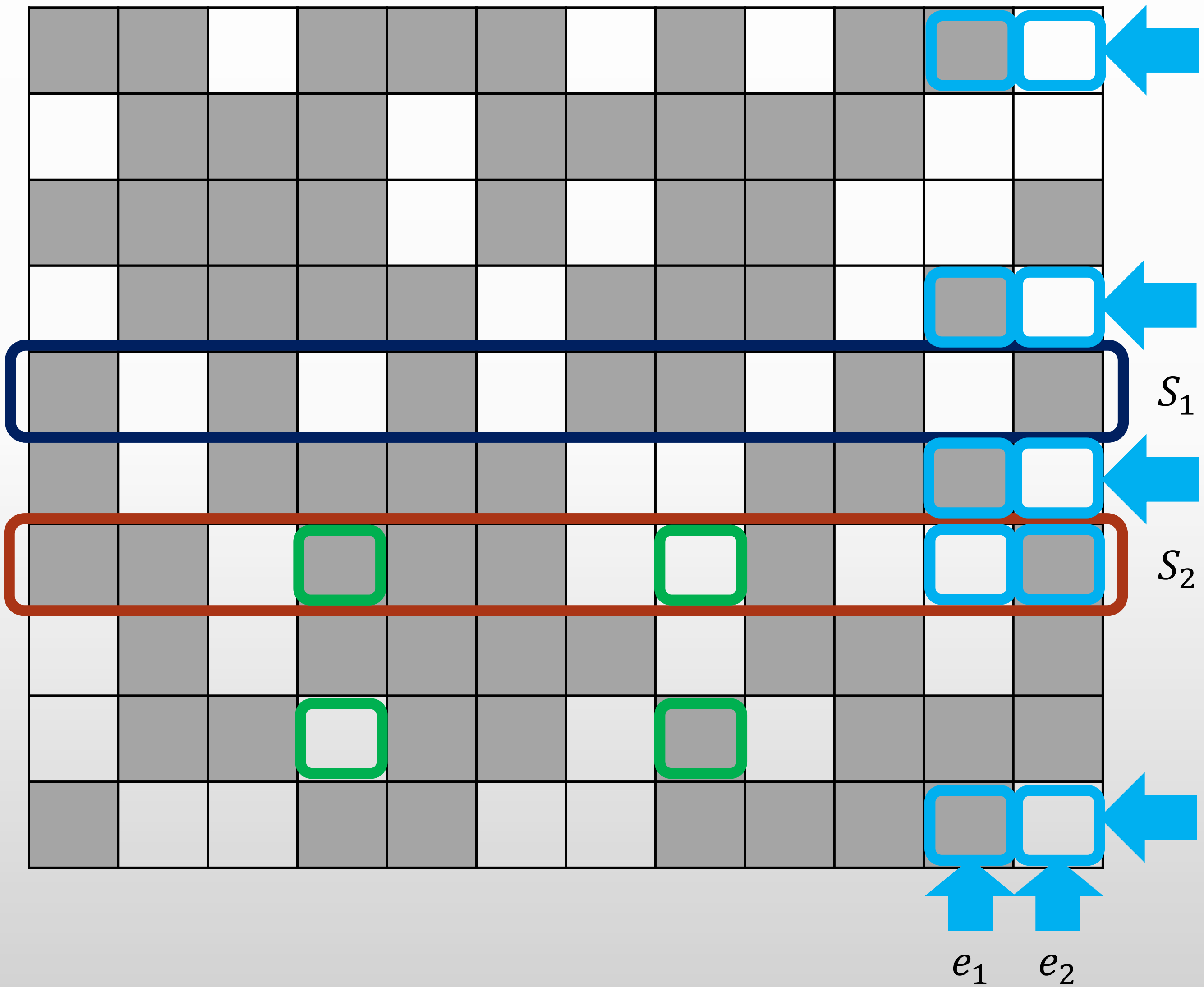
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
    - Find a candidate set
    - **swap**



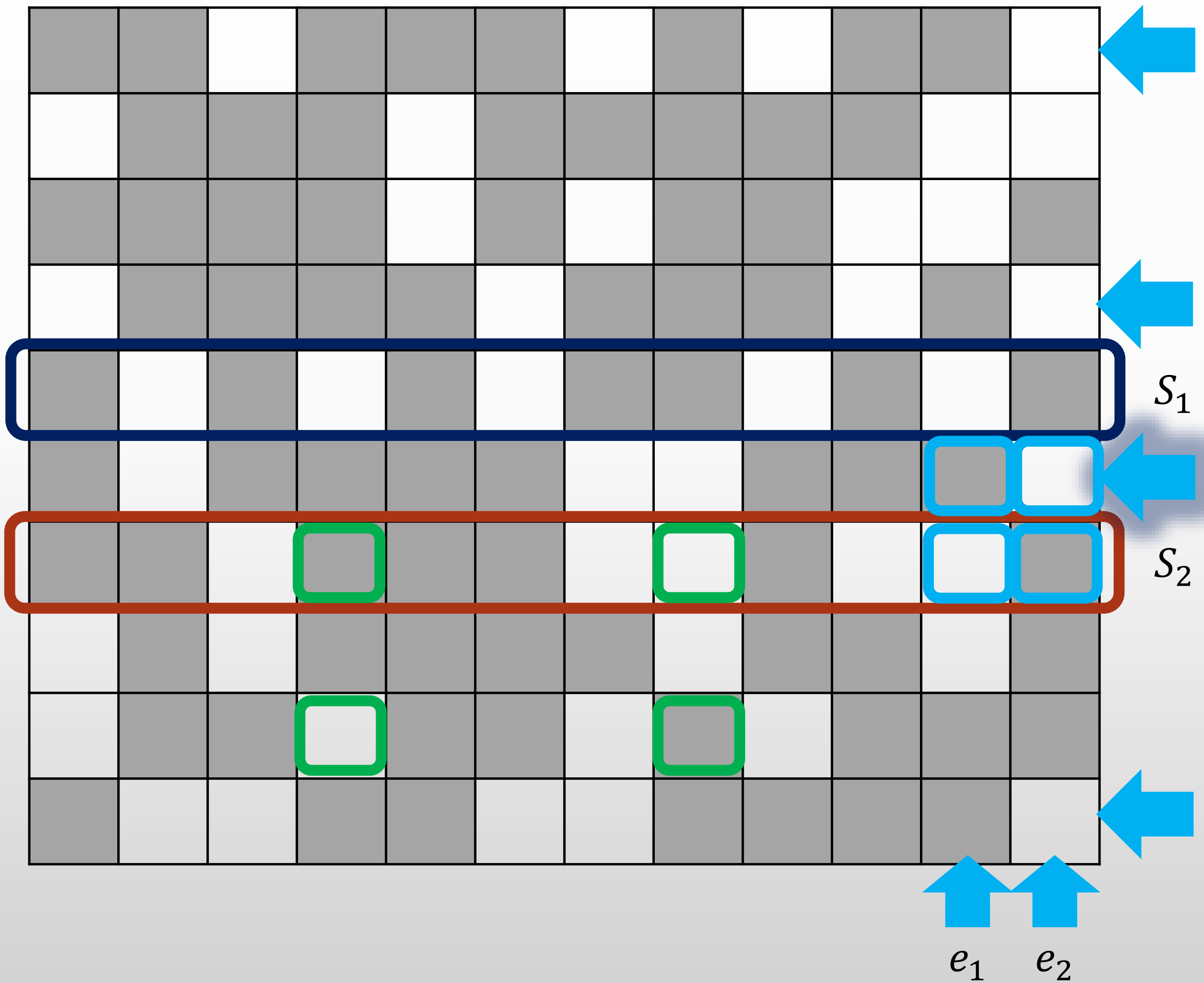
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
    - **Find a candidate set**
    - swap



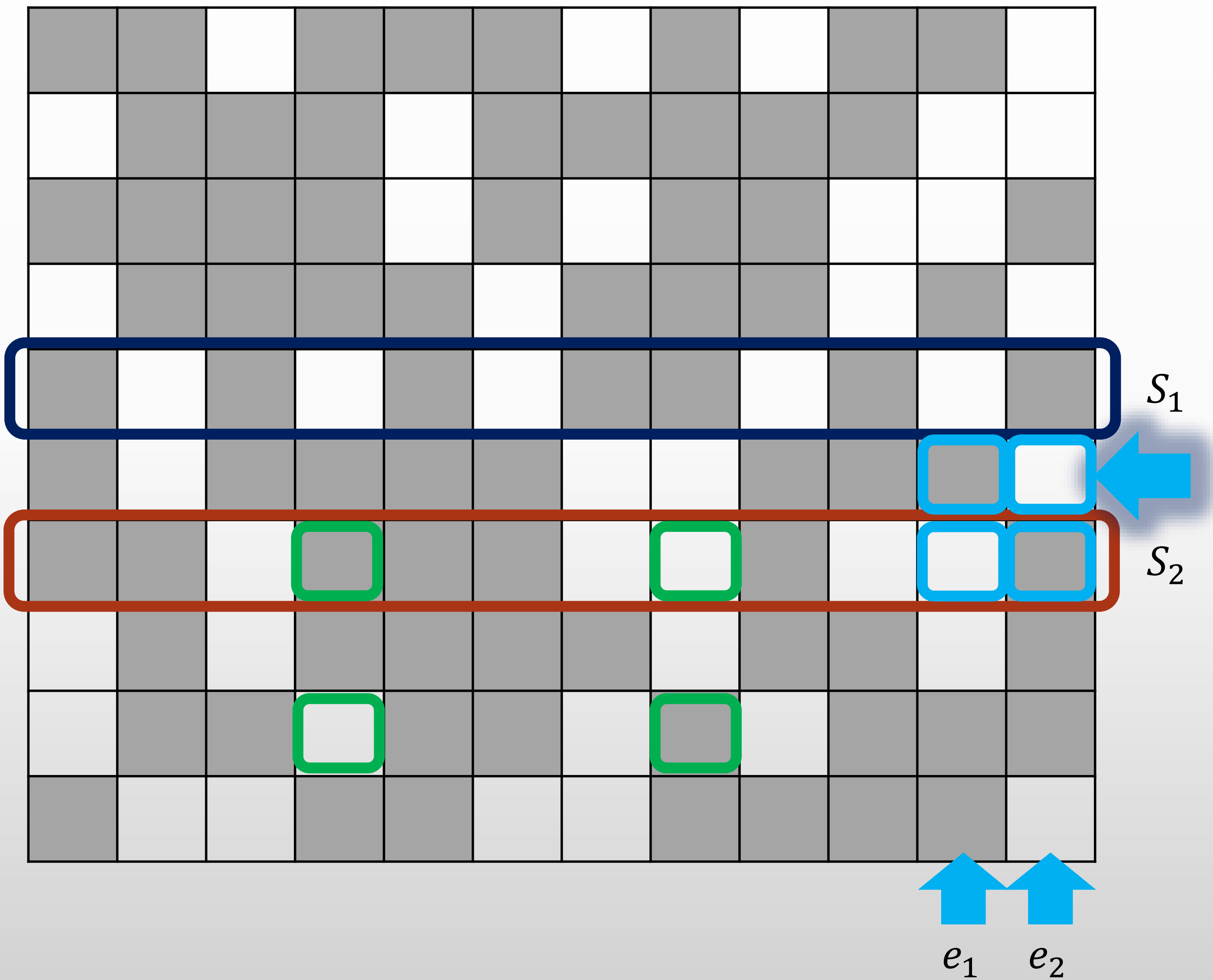
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
  - **Find a candidate set**
  - swap



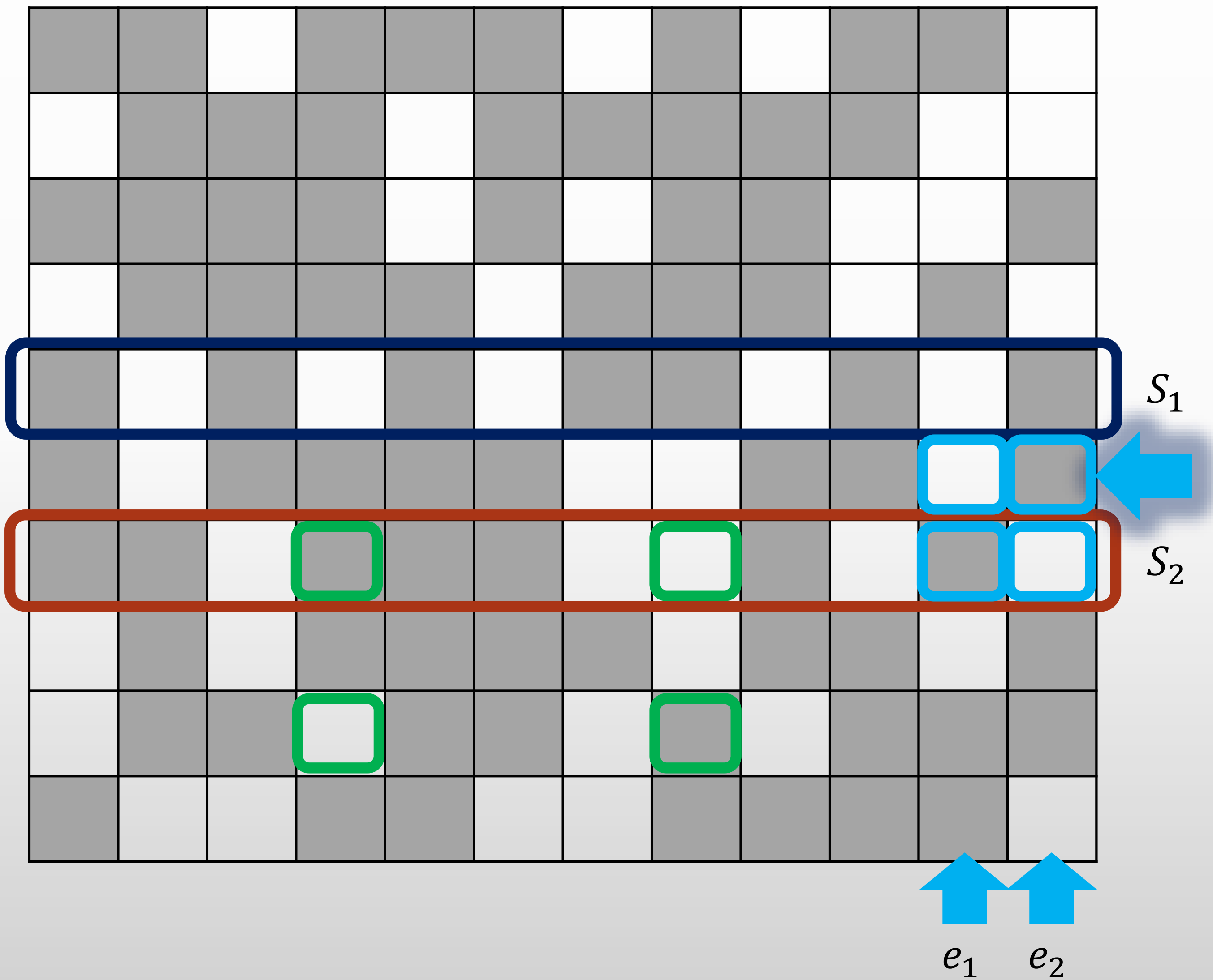
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
    - **Find a candidate set**
    - swap



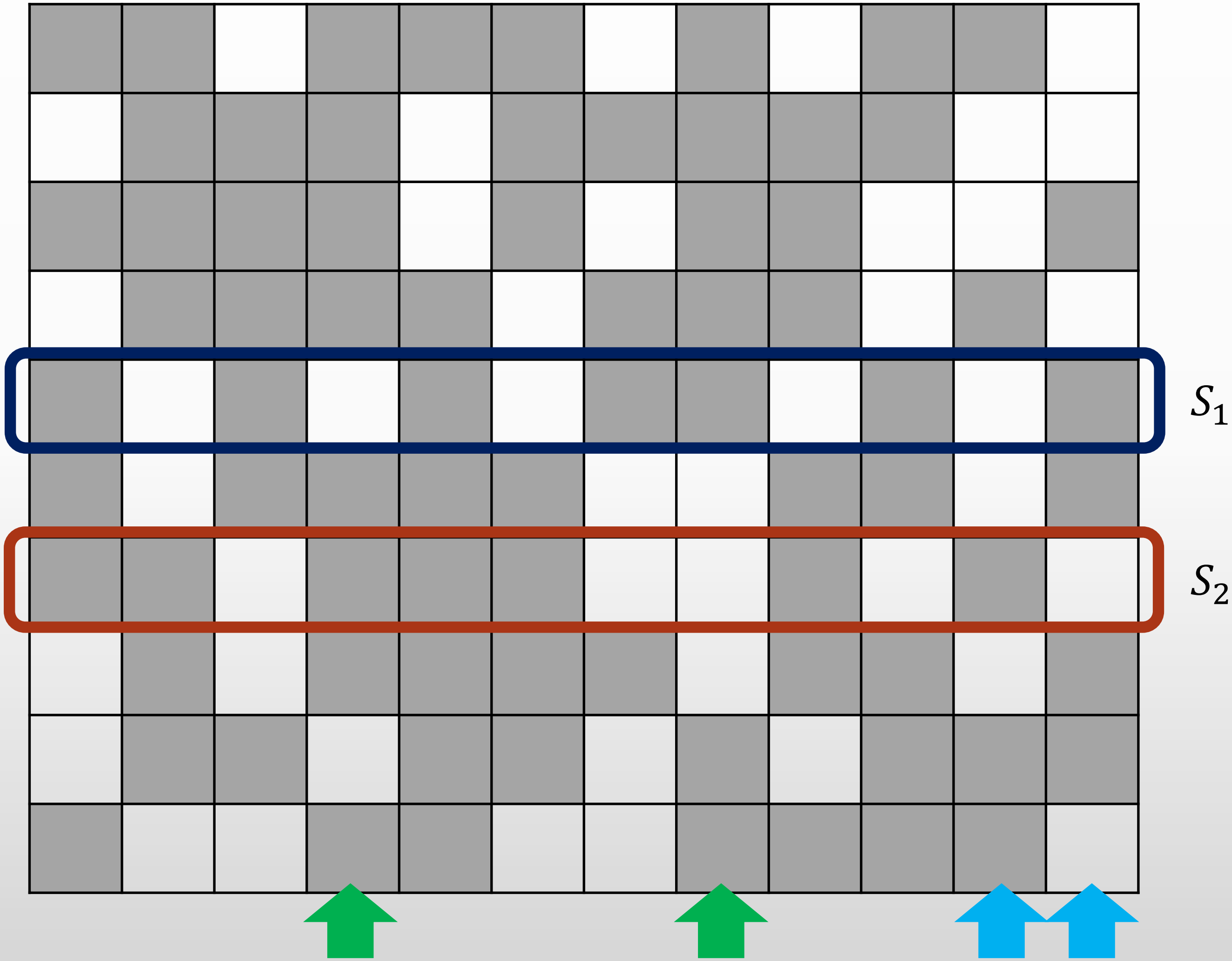
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- **In iteration:**
    - Find a candidate set
    - **swap**



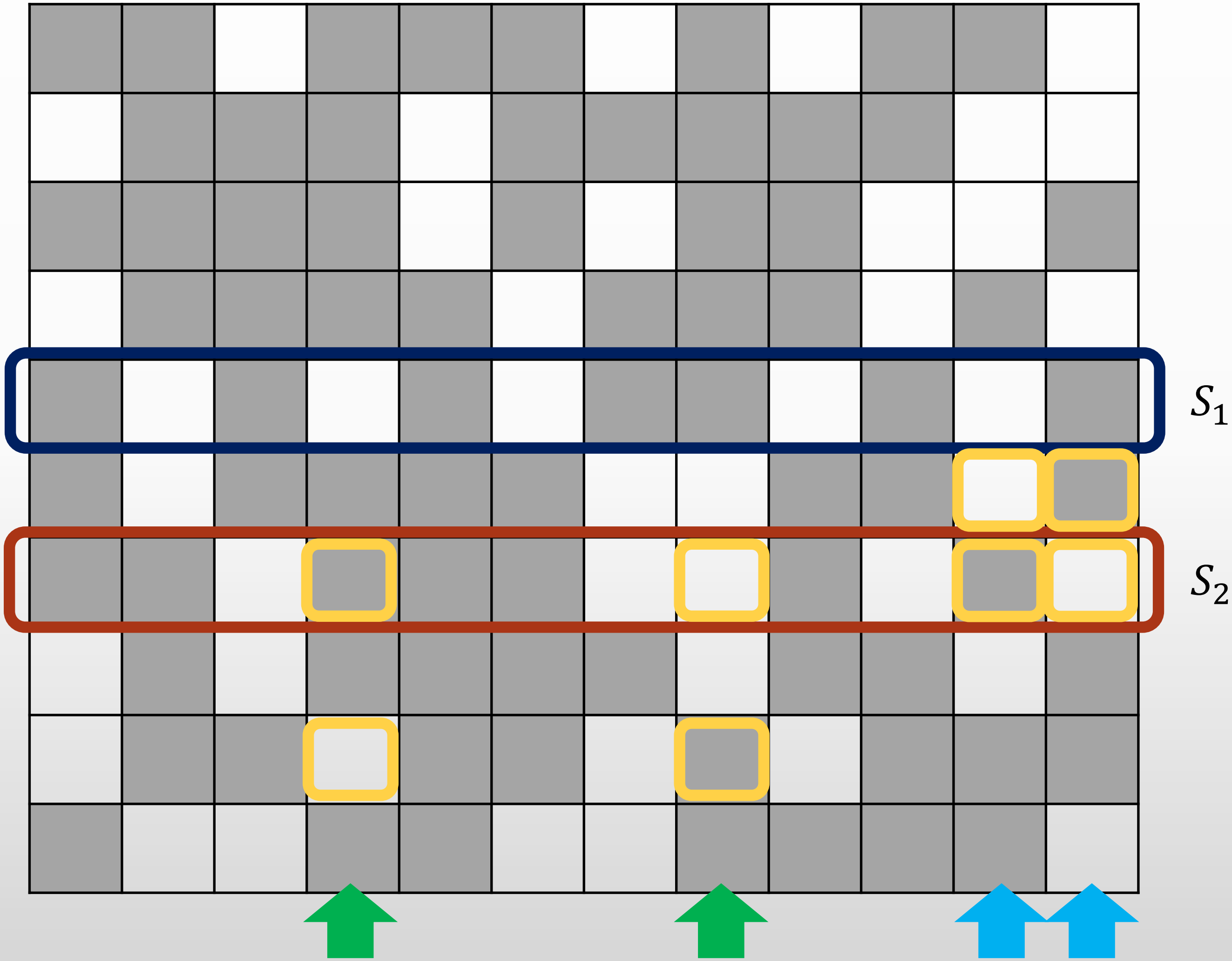
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
  - swap



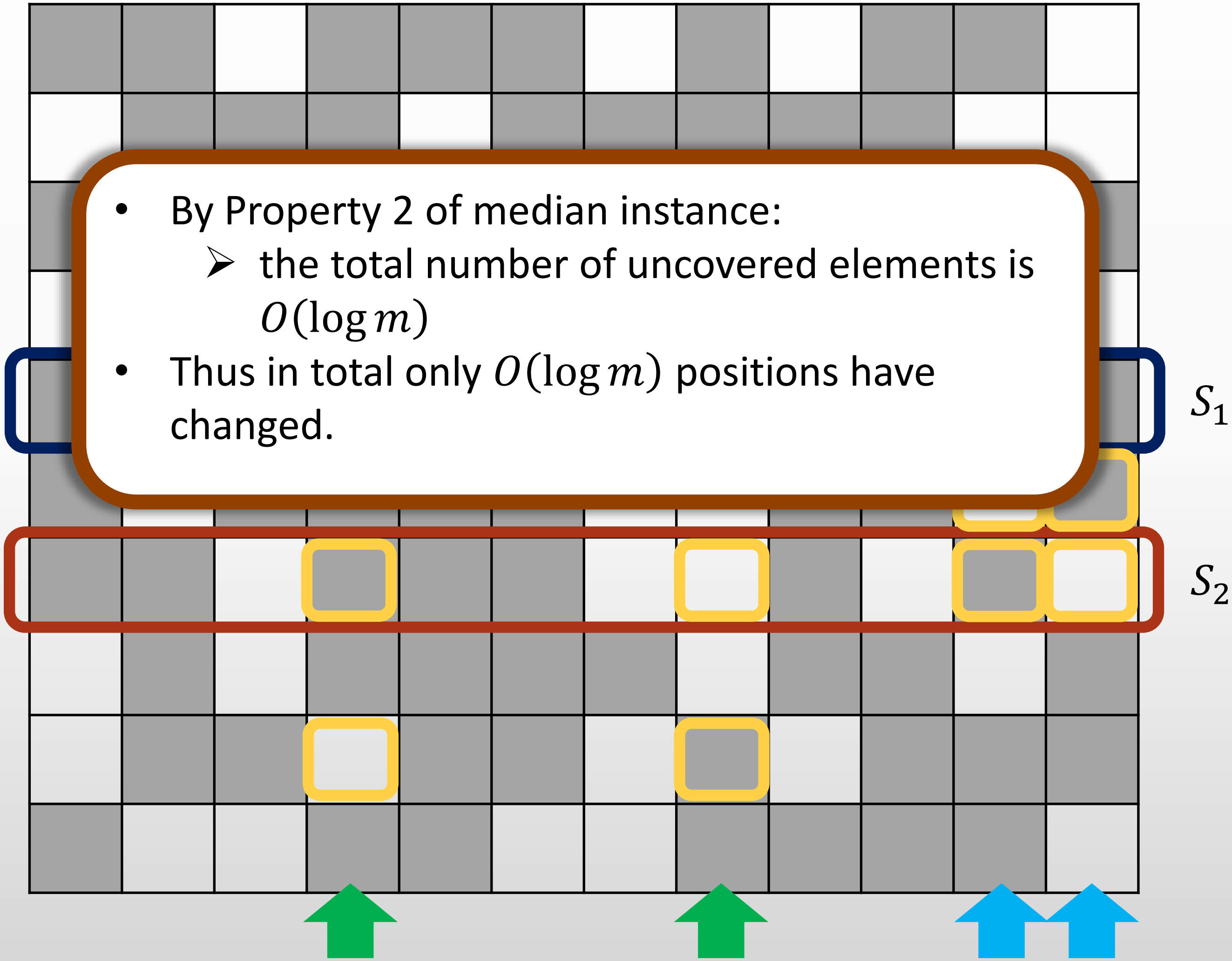
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
    - Find a candidate set
    - swap



The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
  - swap



- By Property 2 of median instance:
  - ➢ the total number of uncovered elements is $O(\log m)$
- Thus in total only $O(\log m)$ positions have changed.

Overall Argument

Lemma: For any element e and any set S, the probability that pair participate in a swap is almost uniform, i.e., $O(\frac{\log m}{mn})$.

- Using other properties of the median instances

Input:

- W.p. ½ the input is the median instance I^*
- W.p. ½ the input is a randomly generated modified instance I

Overall Argument

Lemma: For any element e and any set S, the probability that pair participate in a swap is almost uniform, i.e., $O(\frac{\log m}{mn})$.

- Using other properties of the median instances

Input:

- W.p. ½ the input is the median instance I^*
- W.p. ½ the input is a randomly generated modified instance I

Theorem: Any randomized algorithm that with probability at least 2/3 distinguishes whether the minimum Set Cover size is 2 or at least 3 requires $\widetilde{\boldsymbol{\Omega}}(\boldsymbol{mn})$ number of queries.

Open Problems

Problem	Approximation	Query Complexity	Constraints
Set Cover	$\alpha\rho + 1$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$	$\alpha \geq 2$
	$\rho + 1$	$\tilde{O}\left(\frac{mn}{k}\right)$	–
	α	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$
	α	$\tilde{\Omega}\left(\frac{mn}{k}\right)$	$\alpha \leq 1.01$ $k = O(n/\log m)$
Cover Verification	–	$\tilde{\Omega}(nk)$	$k \leq n/2$

- Prove a lower bound of $\Omega(nk)$ for the set cover problem as well

Open Problems

Problem	Approximation	Query Complexity	Constraints
Set Cover	$\alpha\rho + 1$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$	$\alpha \geq 2$
	$\rho + 1$	$\tilde{O}\left(\frac{mn}{k}\right)$	–
	α	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$
	α	$\tilde{\Omega}\left(\frac{mn}{k}\right)$	$\alpha \leq 1.01$ $k = O(n/\log m)$
Cover Verification	–	$\tilde{\Omega}(nk)$	$k \leq n/2$

- Prove a lower bound of $\Omega(nk)$ for the set cover problem as well
- Similar results for the weighted set cover?

Open Problems

Thank you, Questions?

Problem	Approximation	Query Complexity	
Set Cover	$\alpha\rho + 1$	$\tilde{O}\left(m\left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$	$\alpha \geq 2$
	$\rho + 1$	$\tilde{O}\left(\frac{mn}{k}\right)$	–
	α	$\tilde{\Omega}\left(m\left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$	$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$
	α	$\tilde{\Omega}\left(\frac{mn}{k}\right)$	$\alpha \leq 1.01$ $k = O(n/\log m)$
Cover Verification	–	$\tilde{\Omega}(nk)$	$k \leq n/2$

- Prove a lower bound of $\Omega(nk)$ for the set cover problem as well
- Similar results for the weighted set cover?