Algorithms

Lecture Topic: Approximation Algorithms (Part 3)

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Set Covering Problem"
    1.1 Define "Set Covering Problem".
    1.2 A greedy approximation algorithm for "Set Covering Problem".
    1.3 Analyze the approximation ratio of the algorithm.

The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

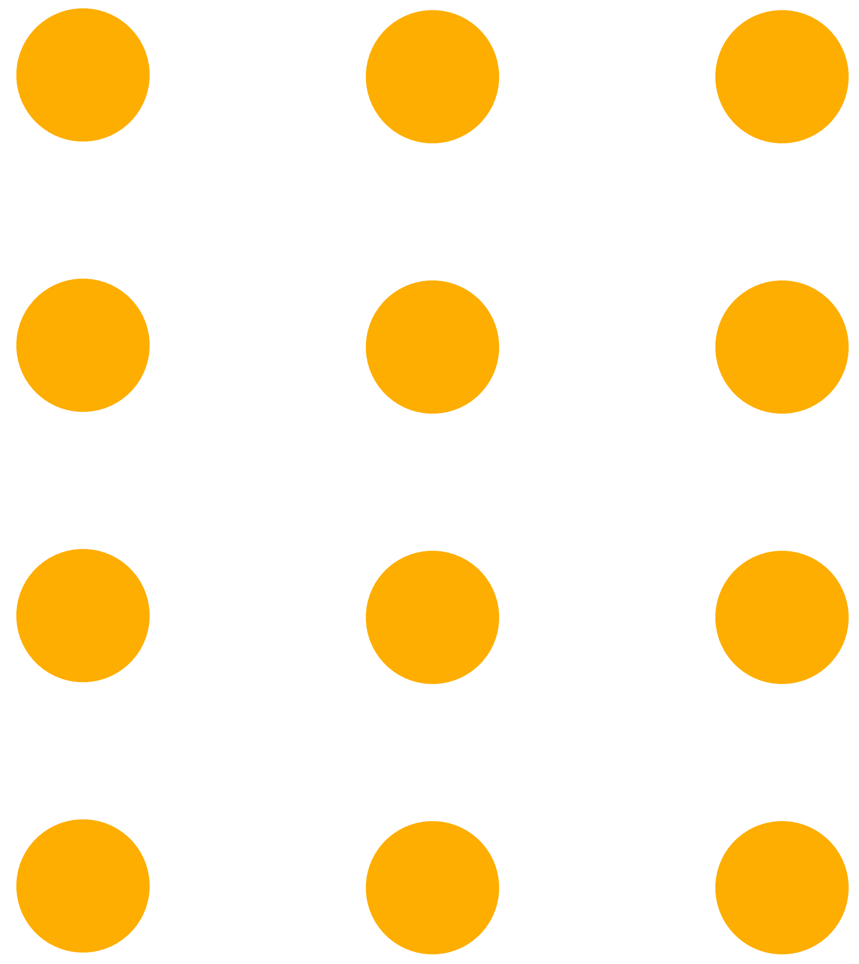
A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

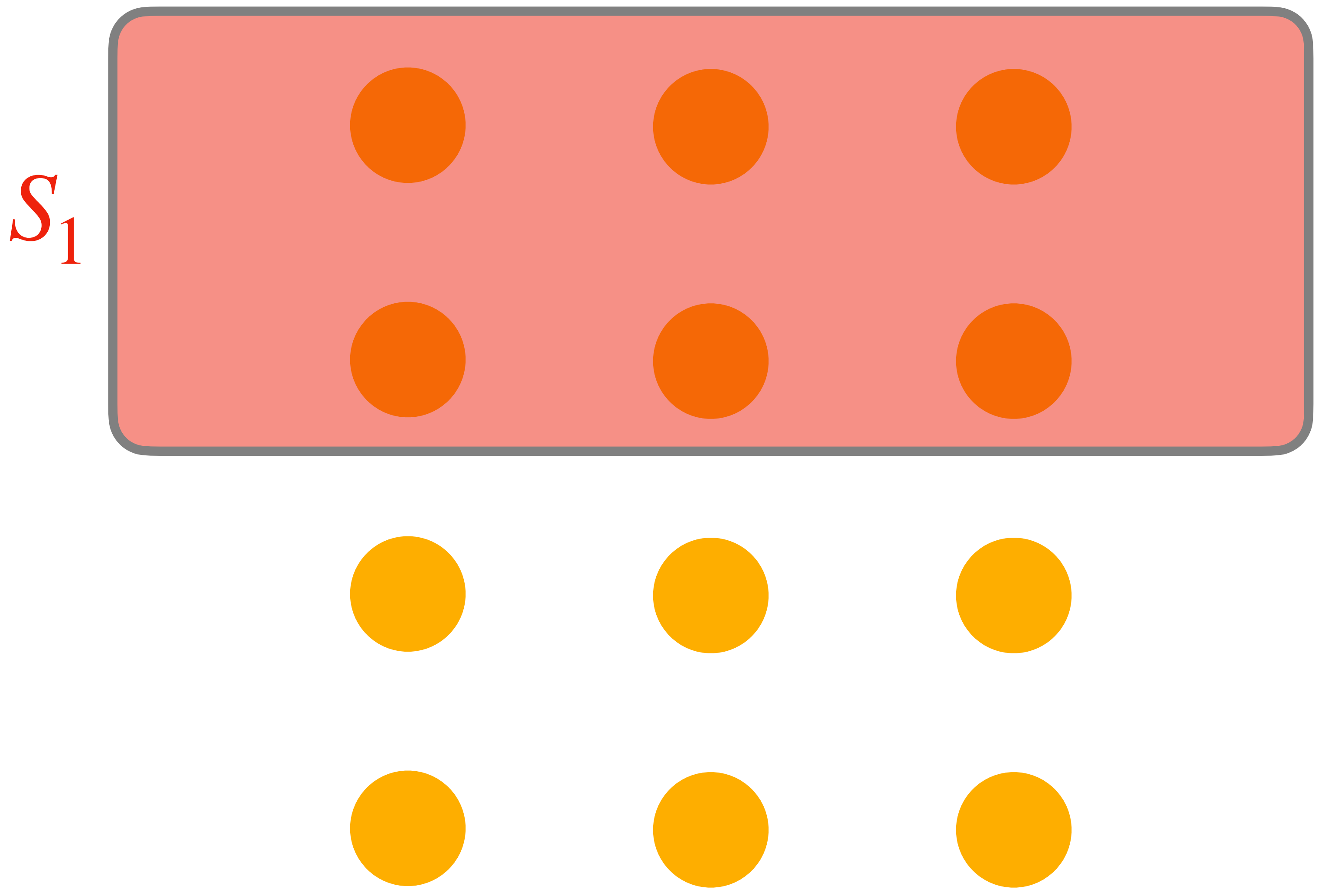
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

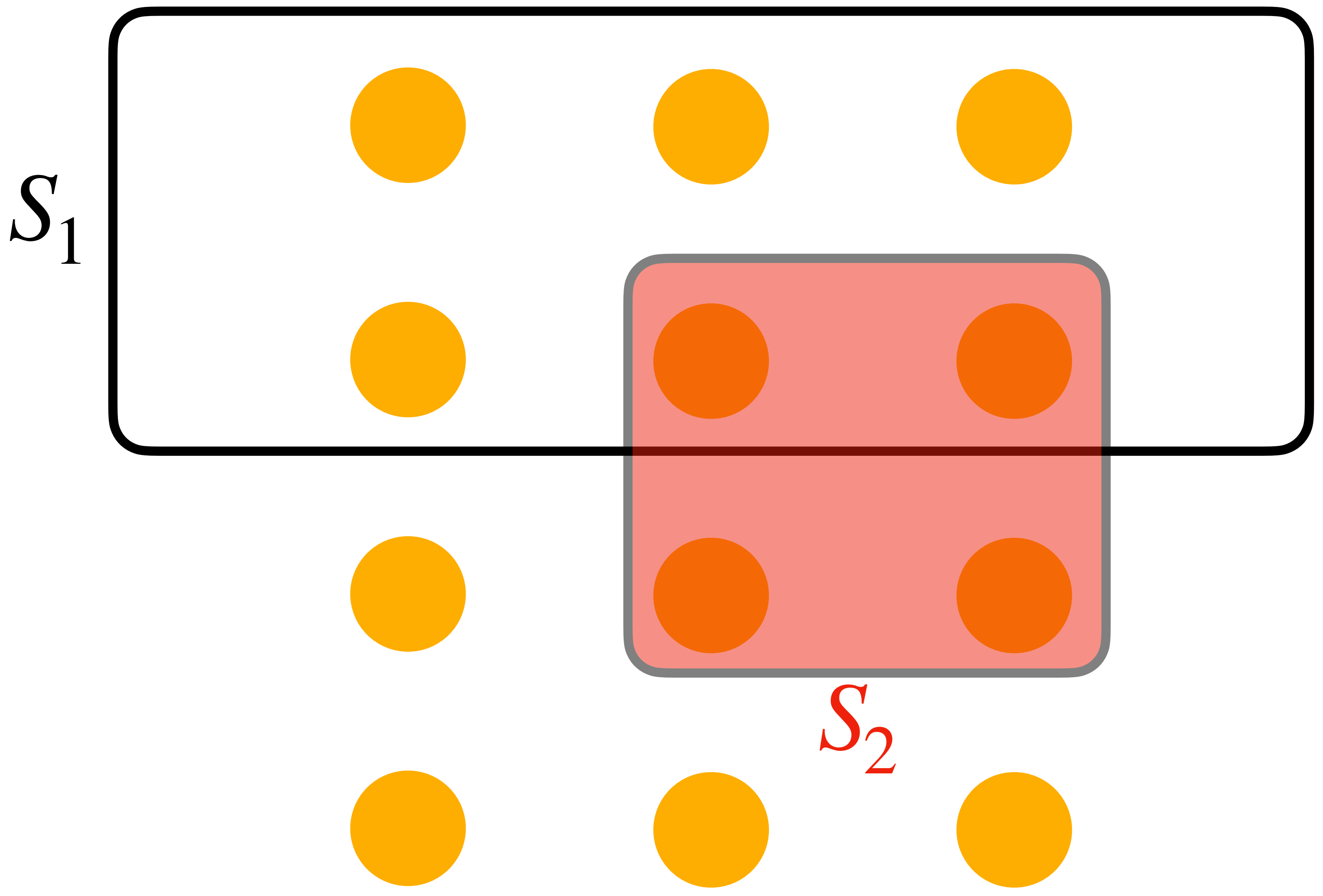
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

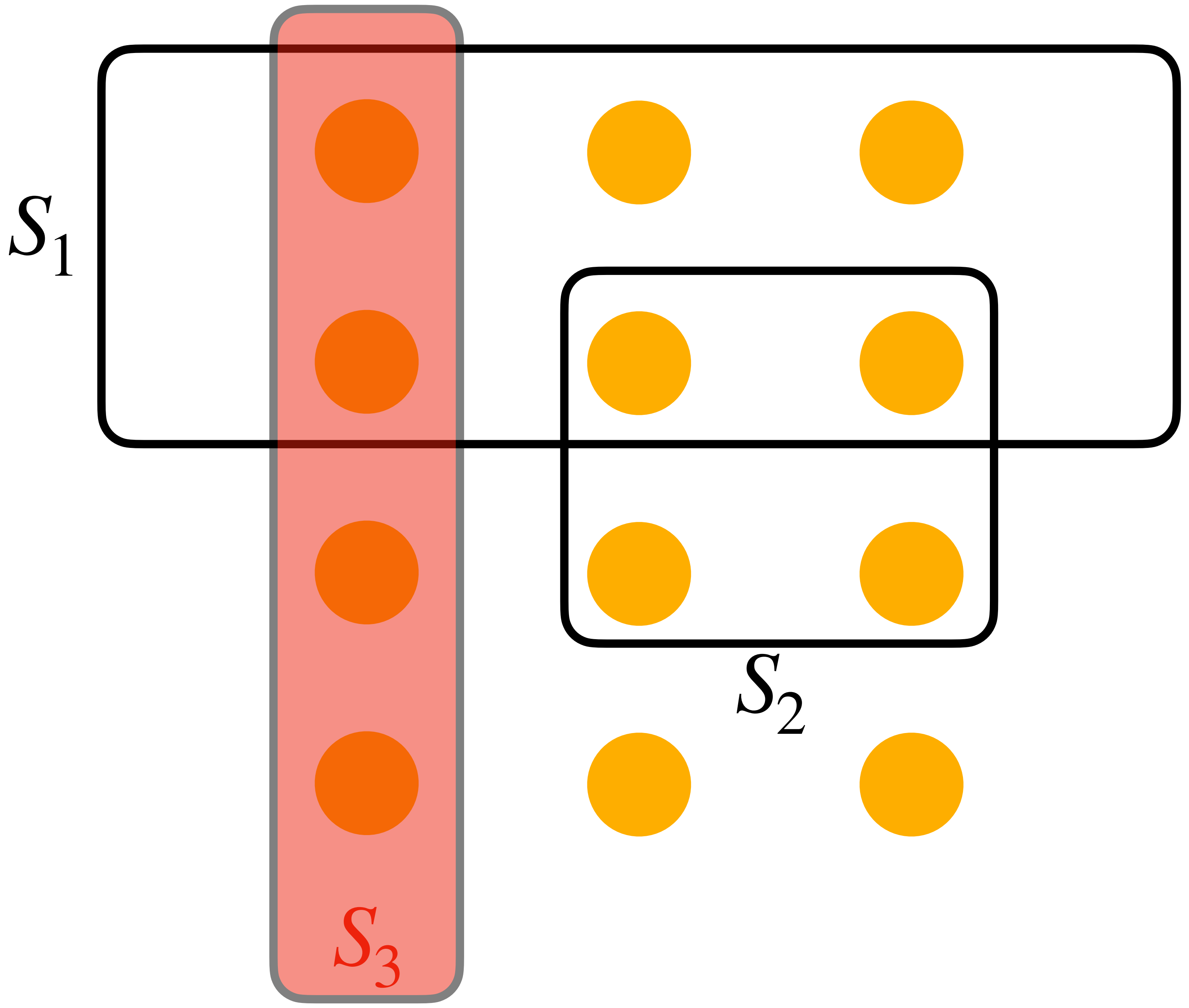
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

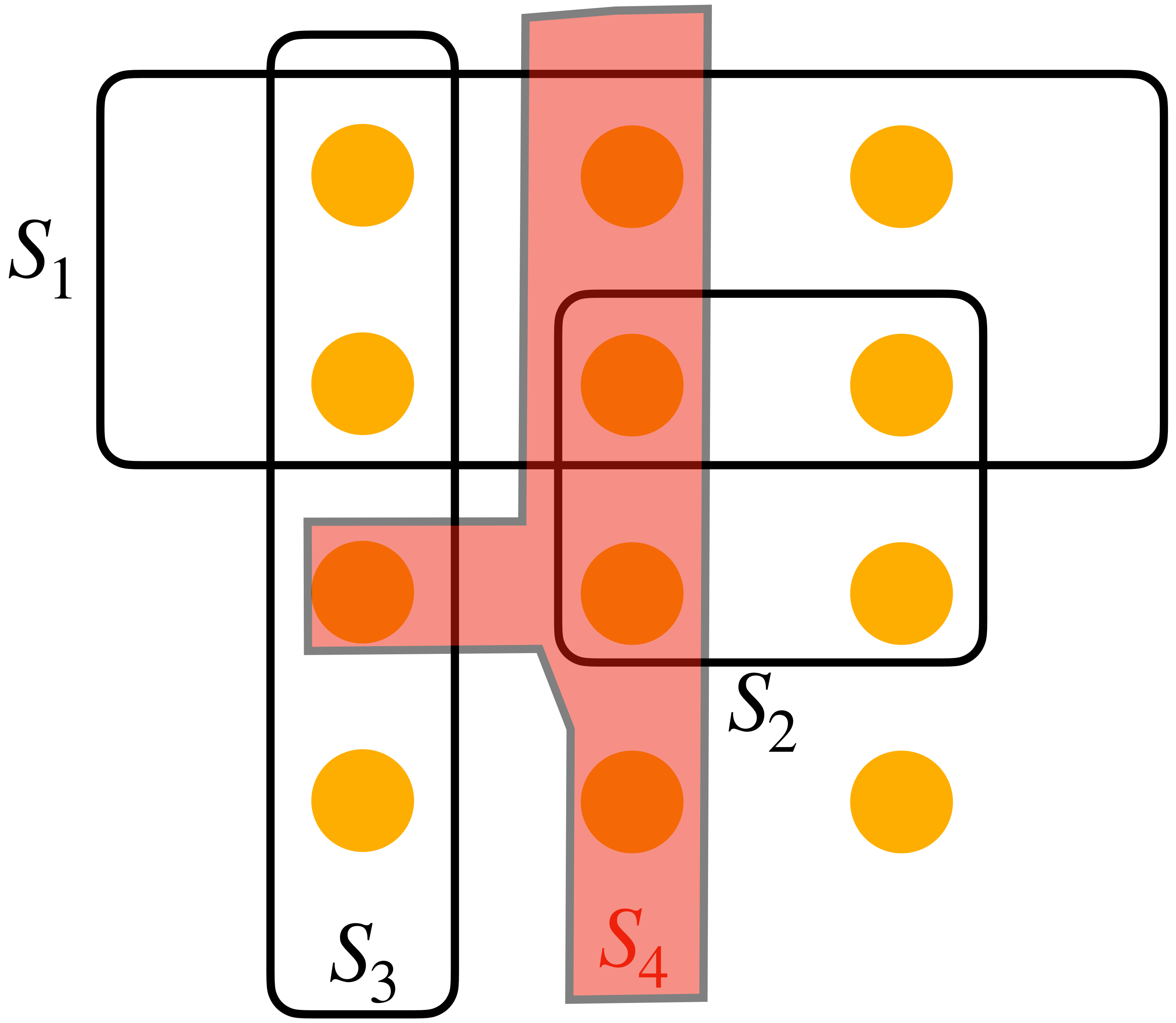
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

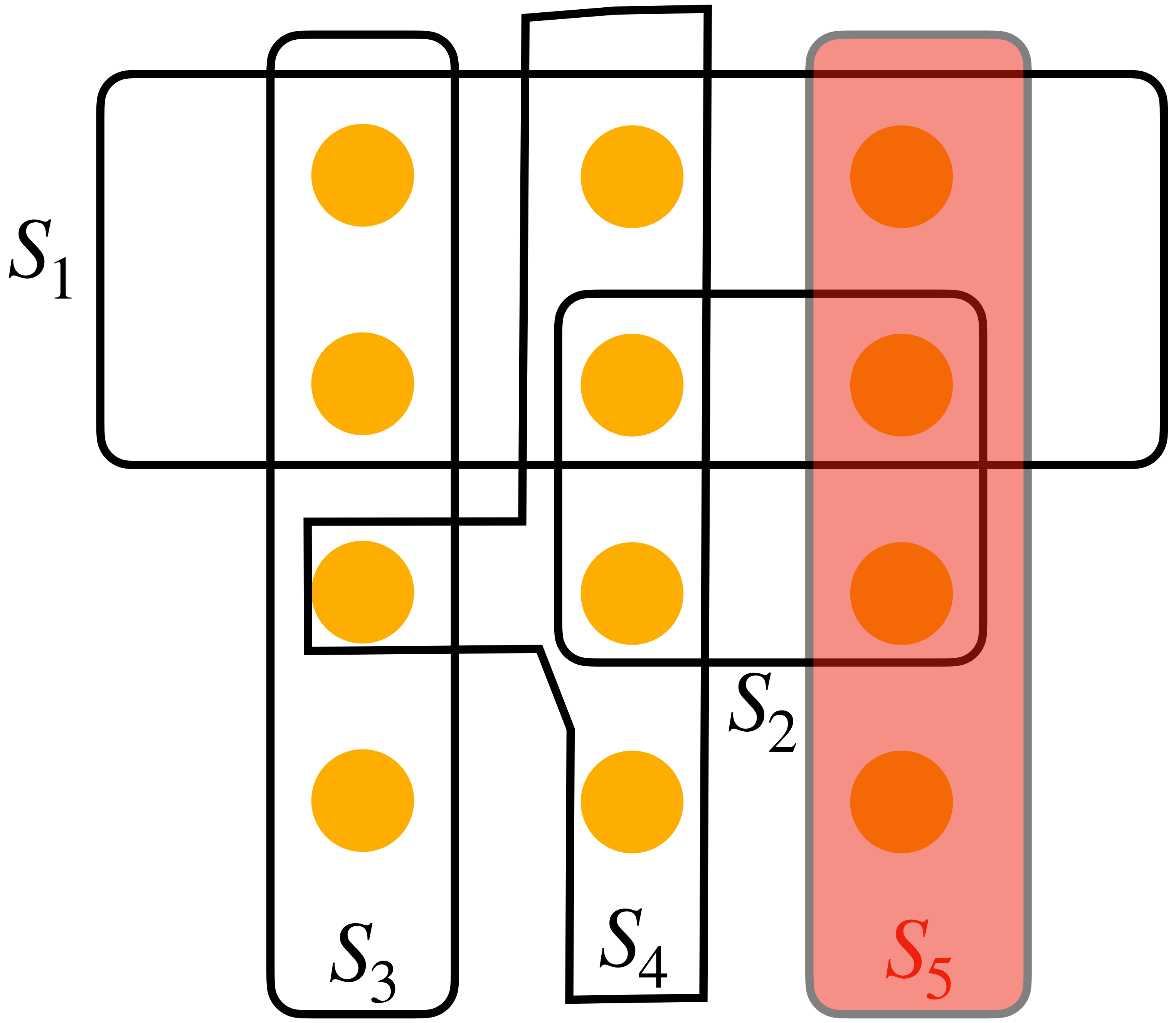
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

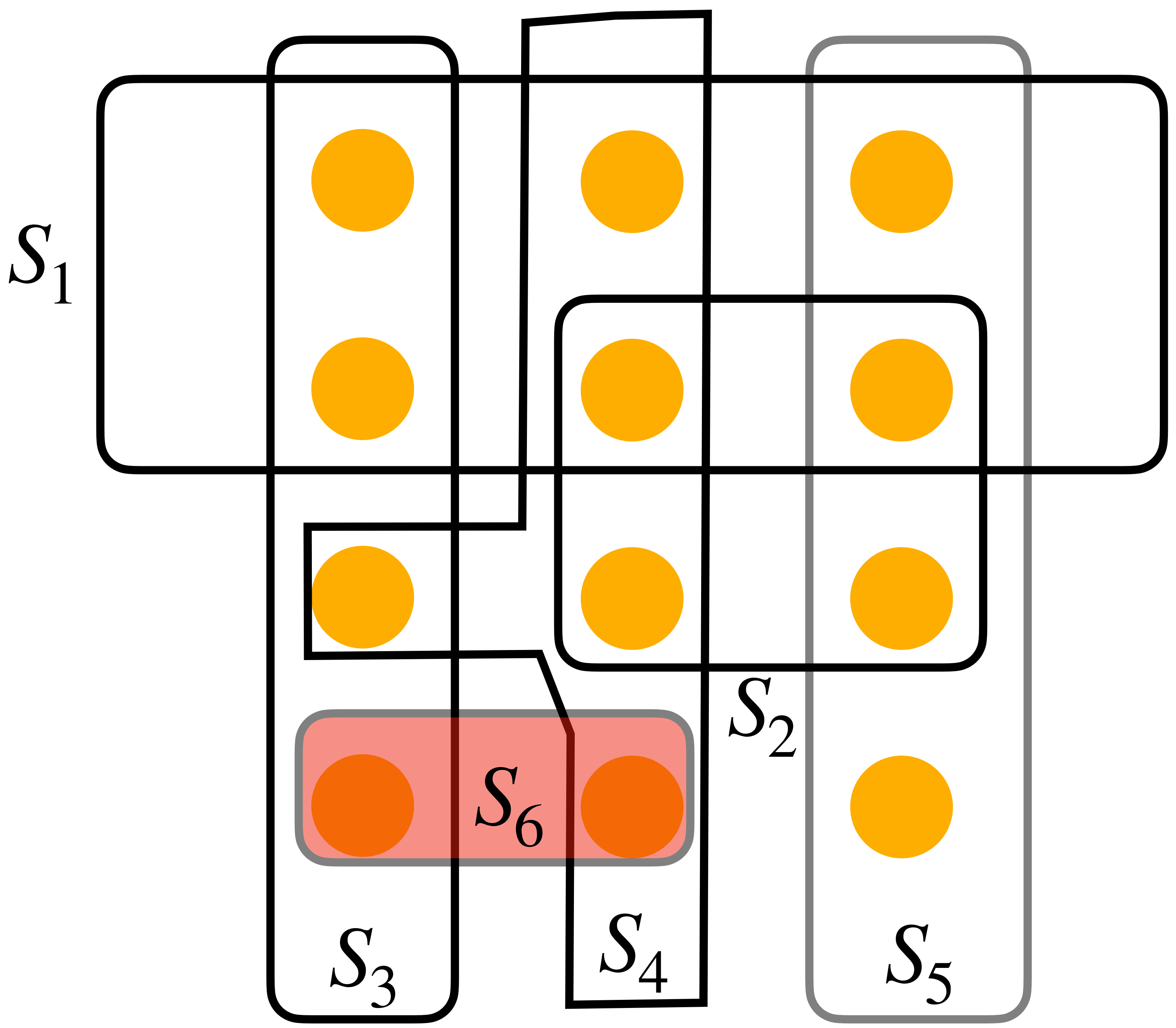
That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$



The Set-Covering Problem

Set-Covering Problem

Input: A set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements.

A family $F = \{S_1, S_2, \cdots, S_m\}$ of m subsets of X, whose union equals X.

That is, $S_i \subseteq X$ for $i = 1,2,\cdots,m$; and $X = \bigcup_{i=1}^{m} S_i$.

Output: A minimum-size subfamily $C \subseteq F$ whose members cover all of X.

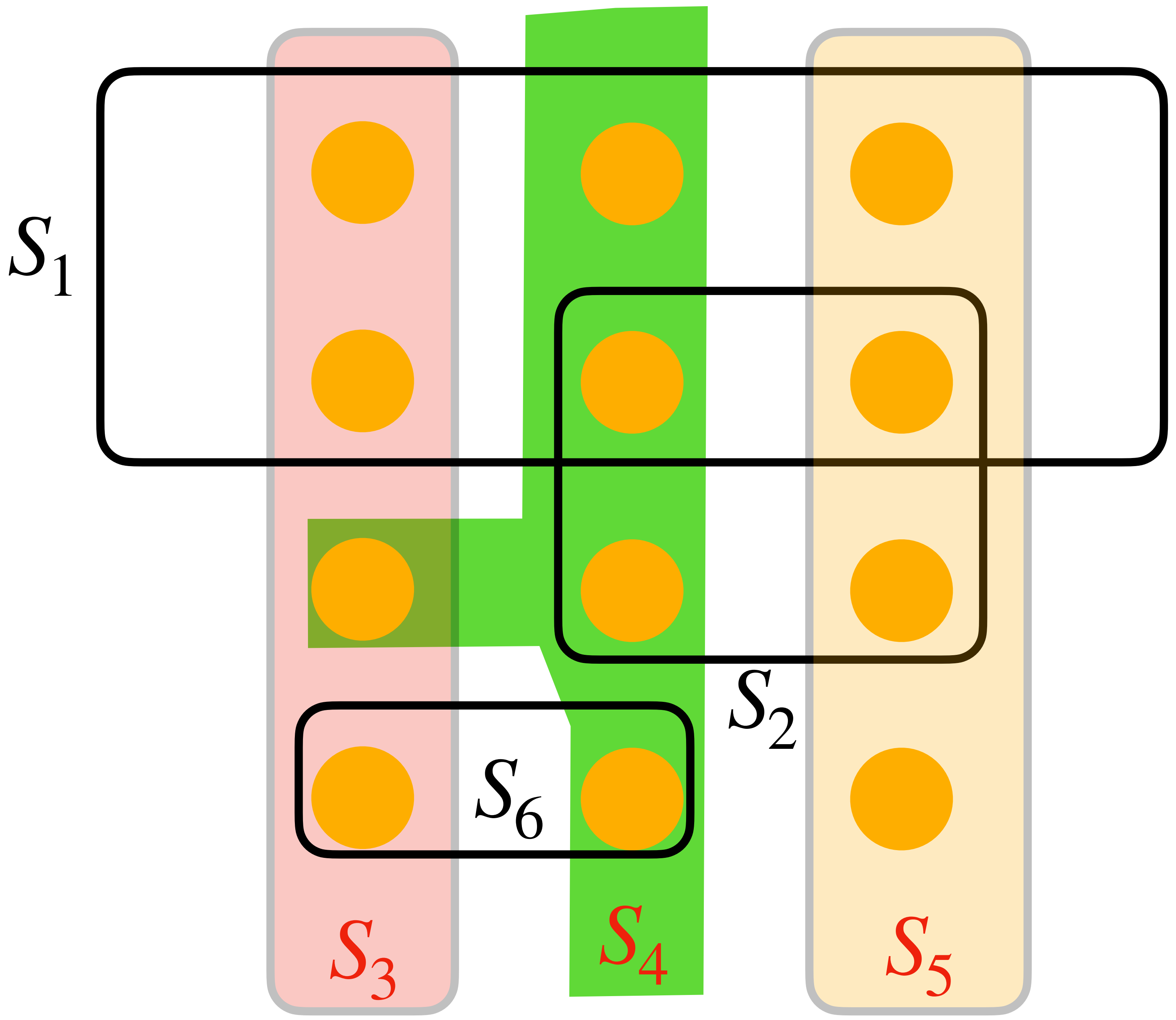
(That is, $X = \bigcup_{S \in C} S$, and $|C|$ is minimized.)

X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

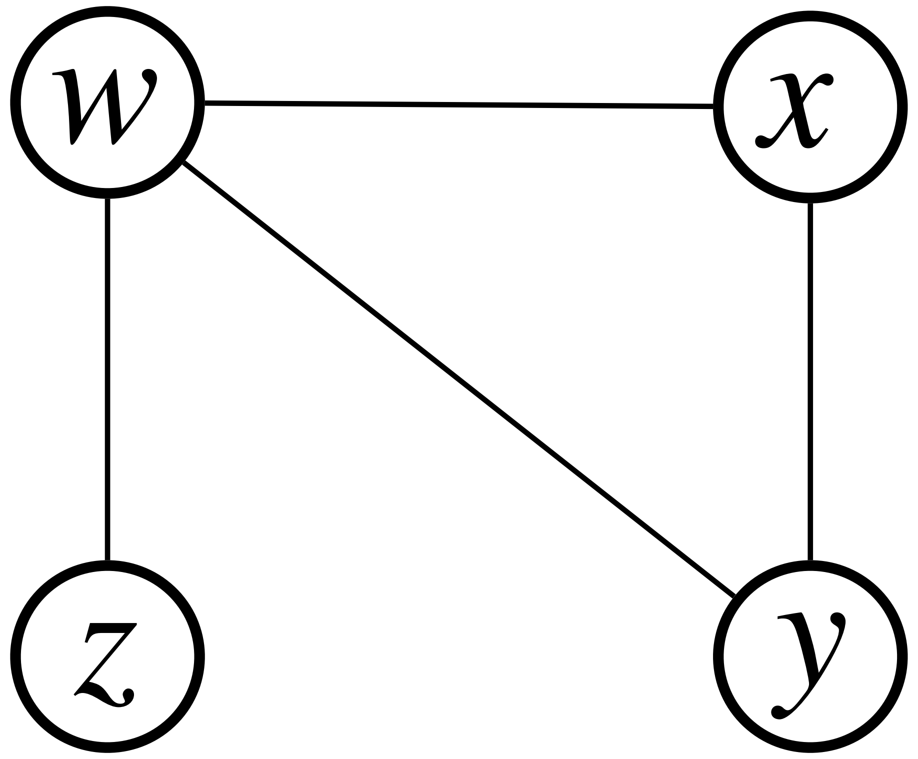
$C = \{S_3, S_4, S_5\}$



The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem

Set-Covering Problem

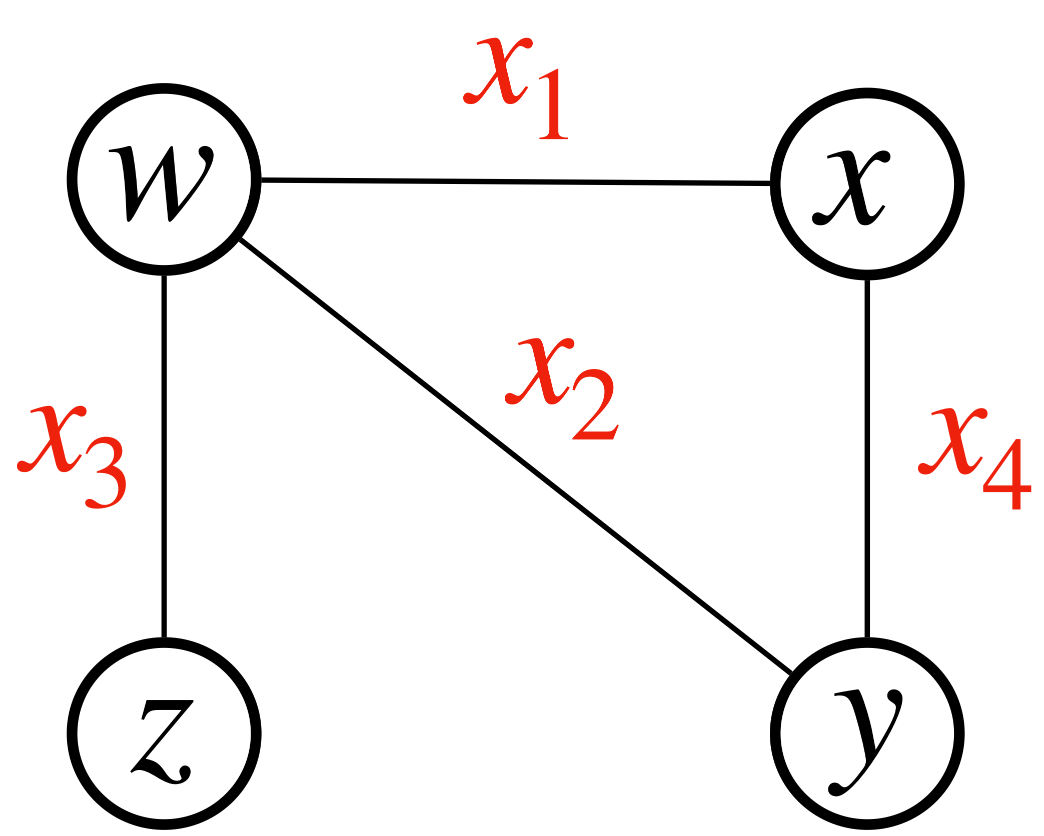


reduction

$k = 2$

The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem



$k = 2$

reduction

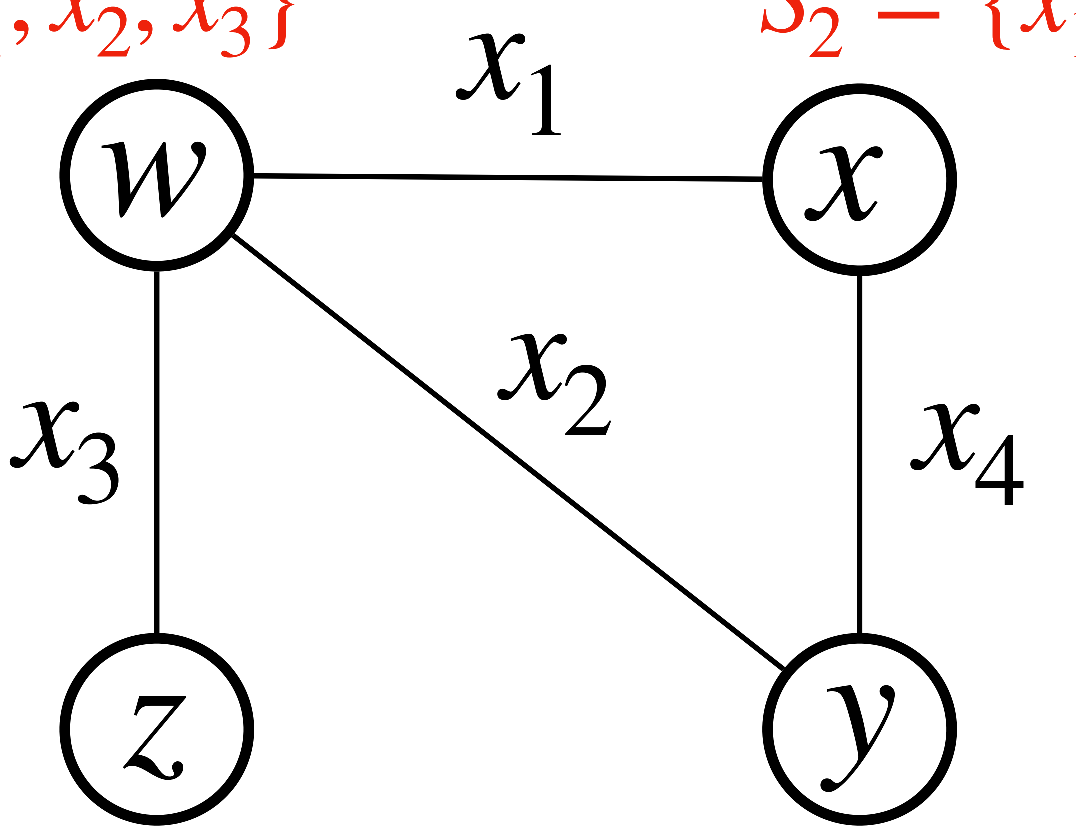
Set-Covering Problem

$X = \{x_1, x_2, x_3, x_4\}$

The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem

$S_1 = \{x_1, x_2, x_3\}$ $\quad$ $S_2 = \{x_1, x_4\}$



$S_4 = \{x_3\}$ $\quad$ $S_3 = \{x_2, x_4\}$

$k = 2$

reduction

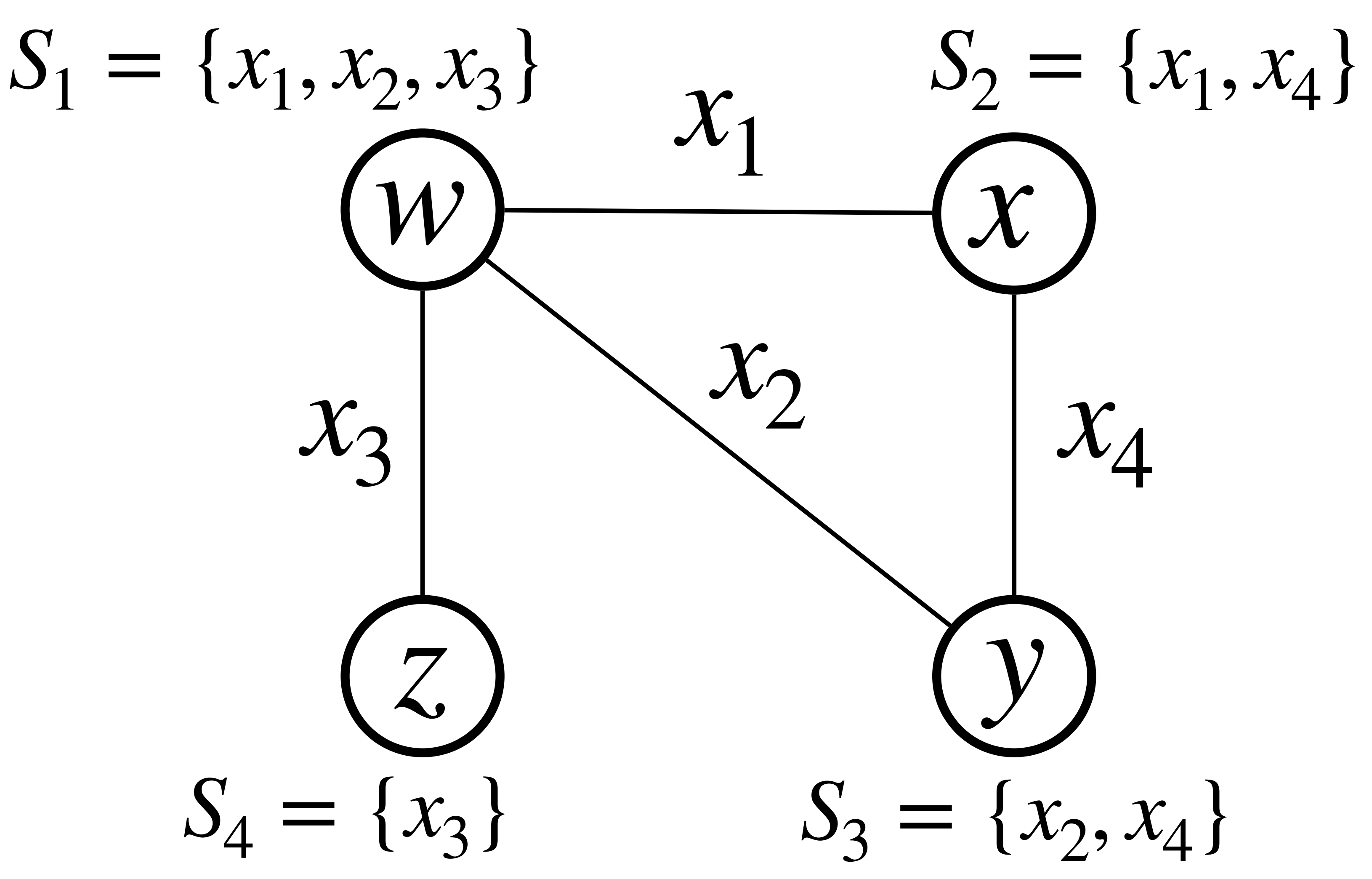
Set-Covering Problem

$$X = \{x_1, x_2, x_3, x_4\}$$

$$F = \{S_1, S_2, S_3, S_4\}$$

The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem



$k = 2$

reduction

Set-Covering Problem

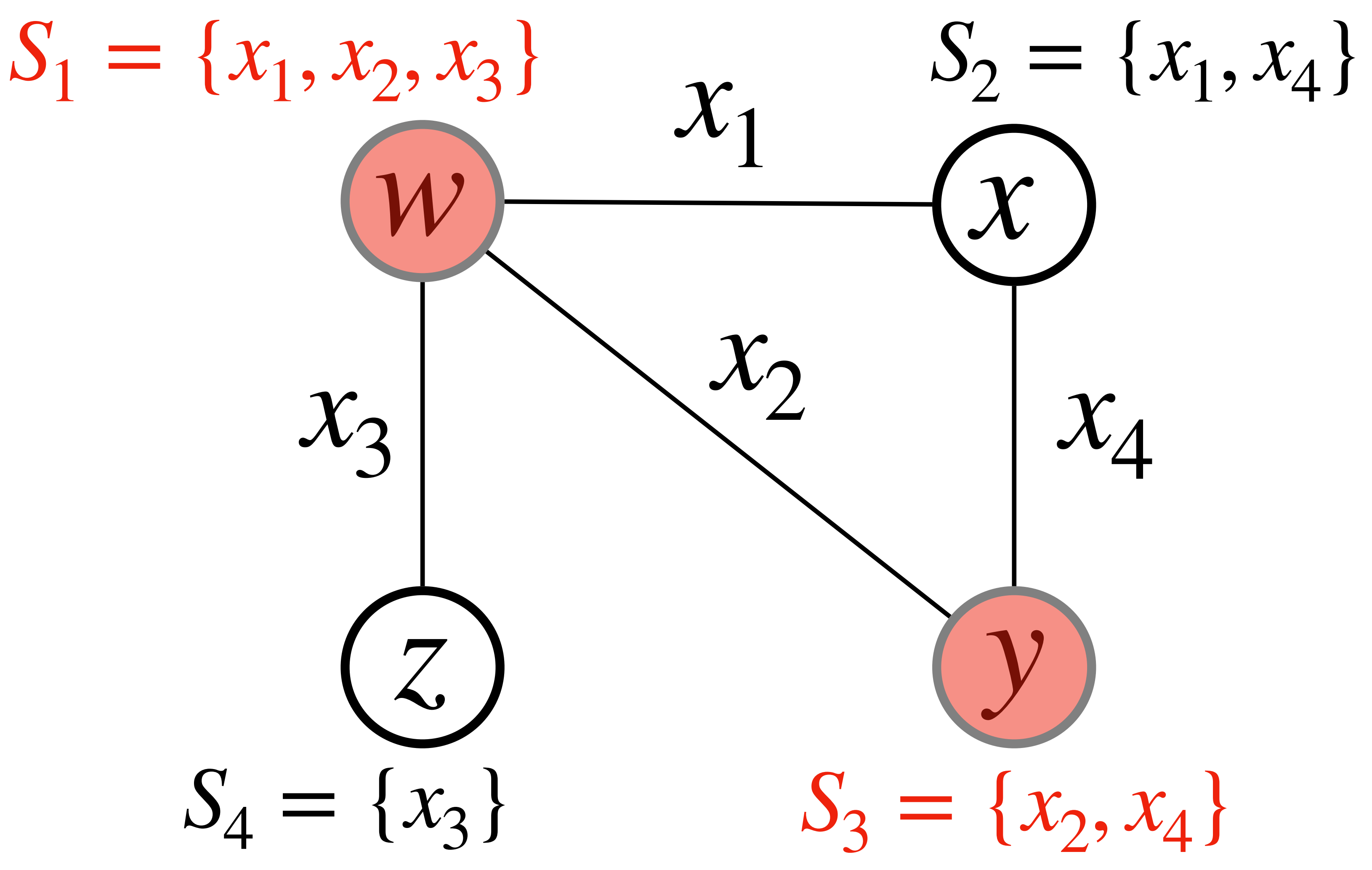
$$X = \{x_1, x_2, x_3, x_4\}$$

$$F = \{S_1, S_2, S_3, S_4\}$$

$$k = 2$$

The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem



$k = 2$

reduction

Set-Covering Problem

$X = \{x_1, x_2, x_3, x_4\}$

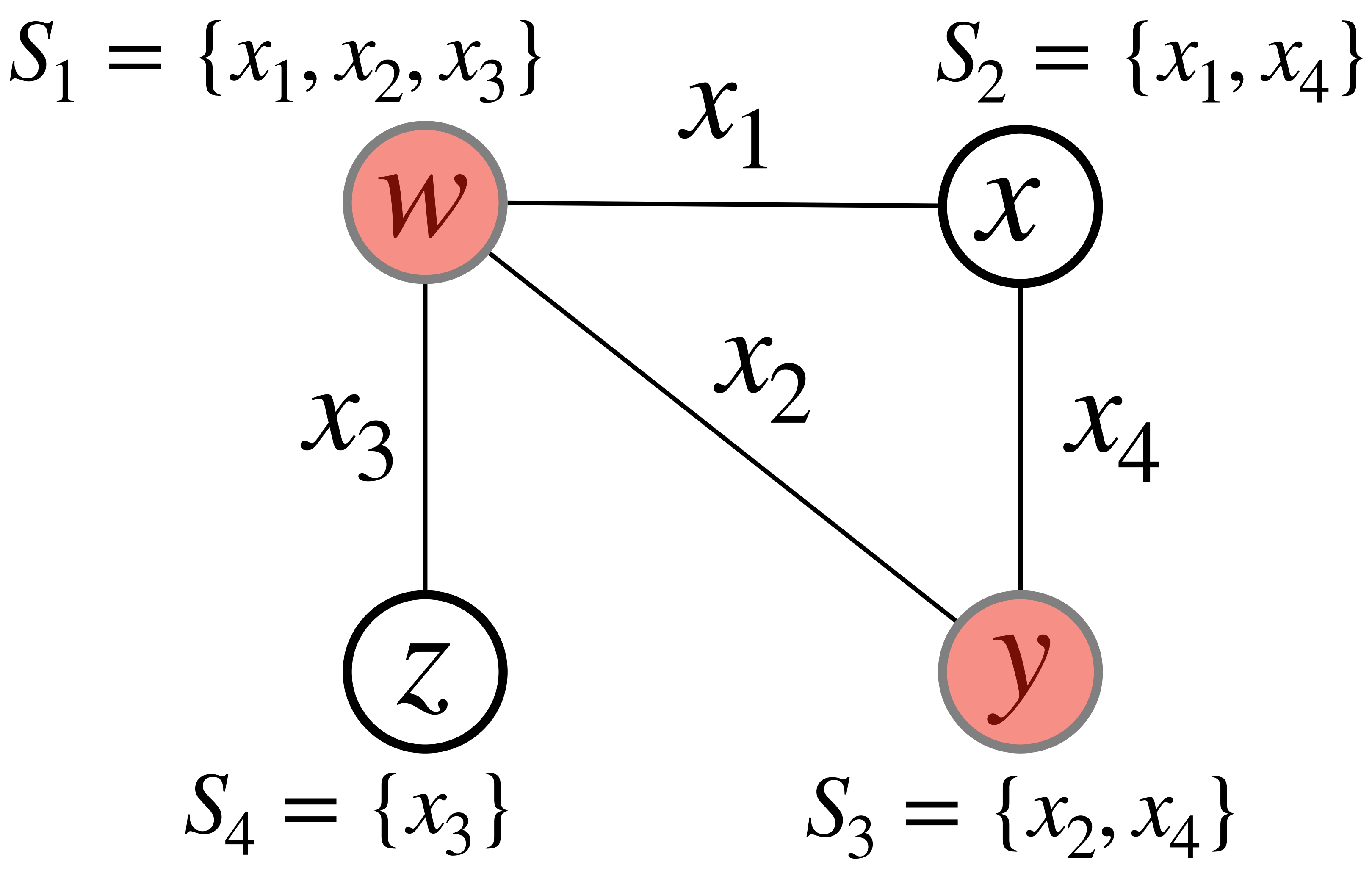
$F = \{S_1, S_2, S_3, S_4\}$

$k = 2$

$C = \{S_1, S_3\}$

The Vertex Cover Problem is a special case of the Set-Covering Problem (as decision problems)

Vertex Cover Problem



$$k = 2$$

reduction

Set-Covering Problem

$$X = \{x_1, x_2, x_3, x_4\}$$

$$F = \{S_1, S_2, S_3, S_4\}$$

$$k = 2$$

$$C = \{S_1, S_3\}$$

The Set-Covering Problem (as a decision problem) is NP-complete.

Quiz questions:

1. What is the relation between the "Set Covering Problem" and the "Vertex Cover Problem"?

2. Can we apply an approximation algorithm for "Vertex Cover Problem" to "Set Covering Problem" and get the same approximation ratio?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Set Covering Problem"
   - 1.1 Define "Set Covering Problem".
   - 1.2 A greedy approximation algorithm for "Set Covering Problem".
   - 1.3 Analyze the approximation ratio of the algorithm.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ U_0 : the set of uncovered elements
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ $\quad$ U_0 : the set of uncovered elements
2. $C = \varnothing$ $\quad$ C : the subfamily of selected subsets
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$ — U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$ — U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $|S \cap U_i|$ — S : the subset that covers as many new elements as possible
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$ — U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $|S \cap U_i|$ — S : the subset that covers as many new elements as possible
6. $U_{i+1} = U_i - S$ — U_{i+1} : the set of uncovered elements after $i + 1$ subsets have been chosen
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$ — U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $|S \cap U_i|$ — S : the subset that covers as many new elements as possible
6. $U_{i+1} = U_i - S$ — U_{i+1} : the set of uncovered elements after $i+1$ subsets have been chosen
7. $C = C \cup \{S\}$ — C : the subfamily of selected subsets
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$ — U_0 : the set of uncovered elements
2. $C = \varnothing$ — C : the subfamily of selected subsets
3. $i = 0$ — i : the number of selected subsets
4. while $U_i \neq \varnothing$ — U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $|S \cap U_i|$ — S : the subset that covers as many new elements as possible
6. $U_{i+1} = U_i - S$ — U_{i+1} : the set of uncovered elements after $i+1$ subsets have been chosen
7. $C = C \cup \{S\}$ — C : the subfamily of selected subsets
8. $i = i + 1$ — i : the number of selected subsets
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$	U_0 : the set of uncovered elements
2. $C = \varnothing$	C : the subfamily of selected subsets
3. $i = 0$	i : the number of selected subsets
4. while $U_i \neq \varnothing$	U_i : the set of uncovered elements after i subsets have been chosen
5. select $S \in F$ that maximizes $\vert S \cap U_i \vert$	S : the subset that covers as many new elements as possible
6. $U_{i+1} = U_i - S$	U_{i+1} : the set of uncovered elements after $i+1$ subsets have been chosen
7. $C = C \cup \{S\}$	C : the subfamily of selected subsets
8. $i = i + 1$	i : the number of selected subsets
9. return C	C : the subfamily of selected subsets

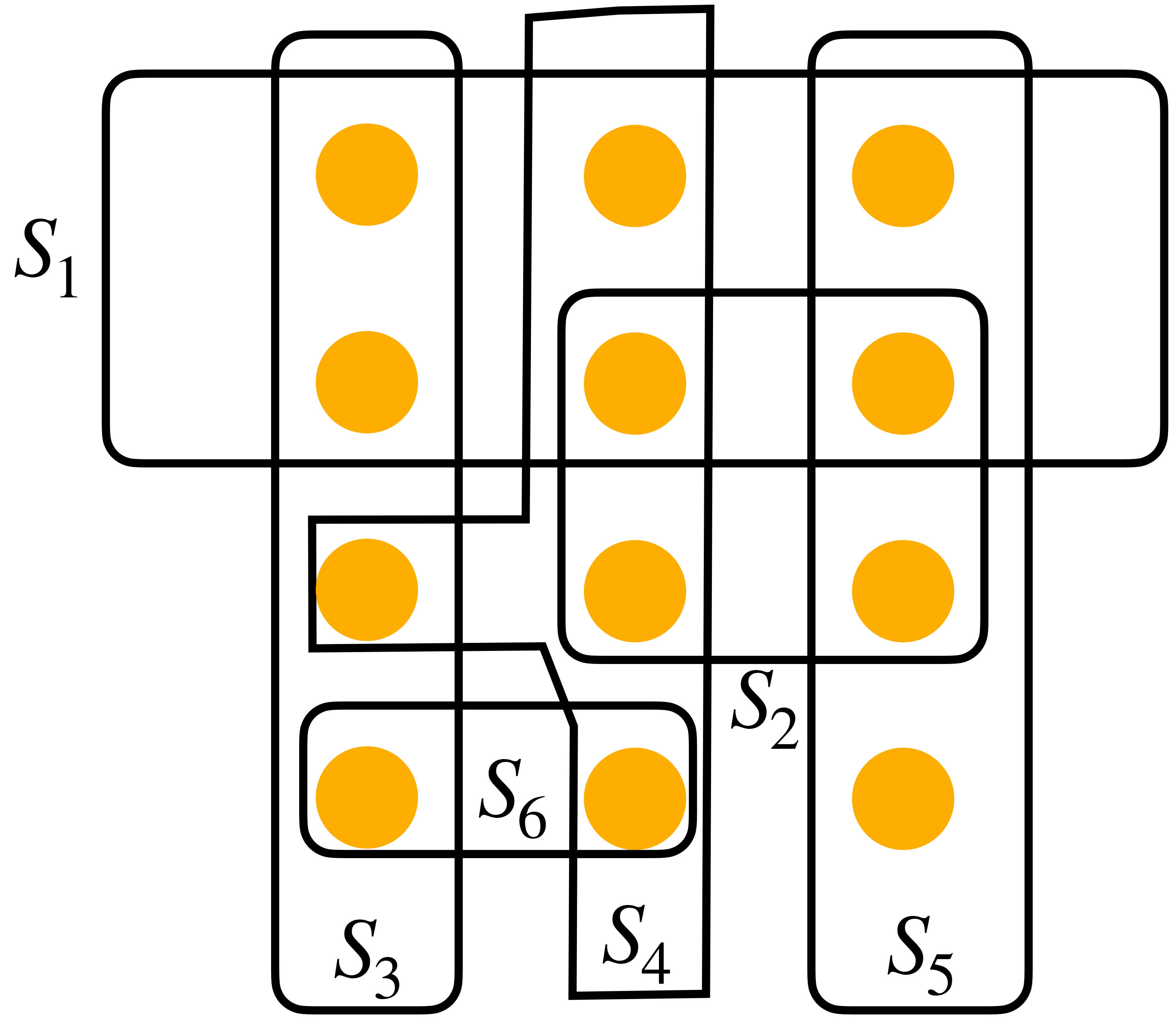
Idea: Each time, pick a subset that covers as many new elements as possible.

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.



X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

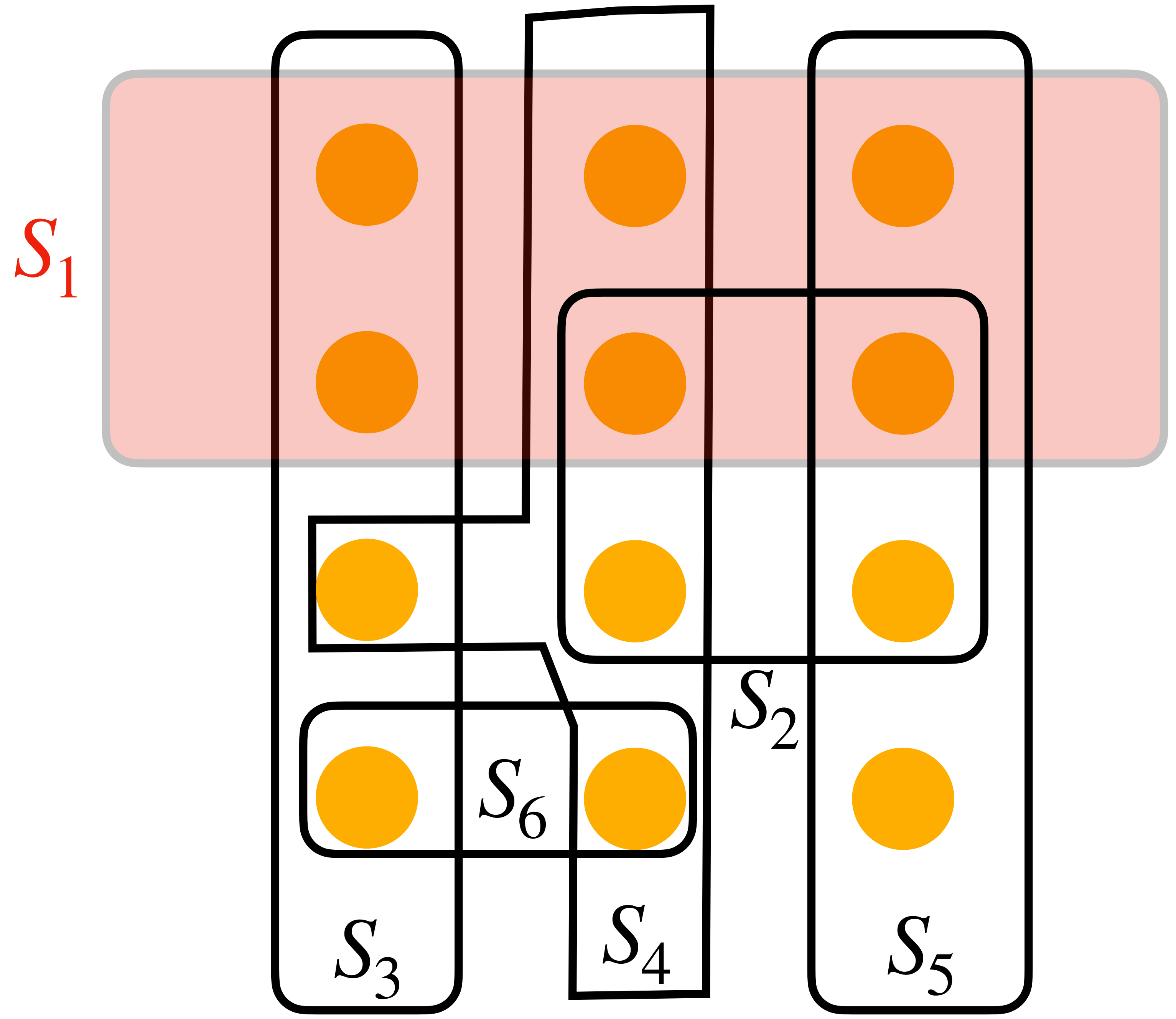
$C = \{S_3, S_4, S_5\}$

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.



X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

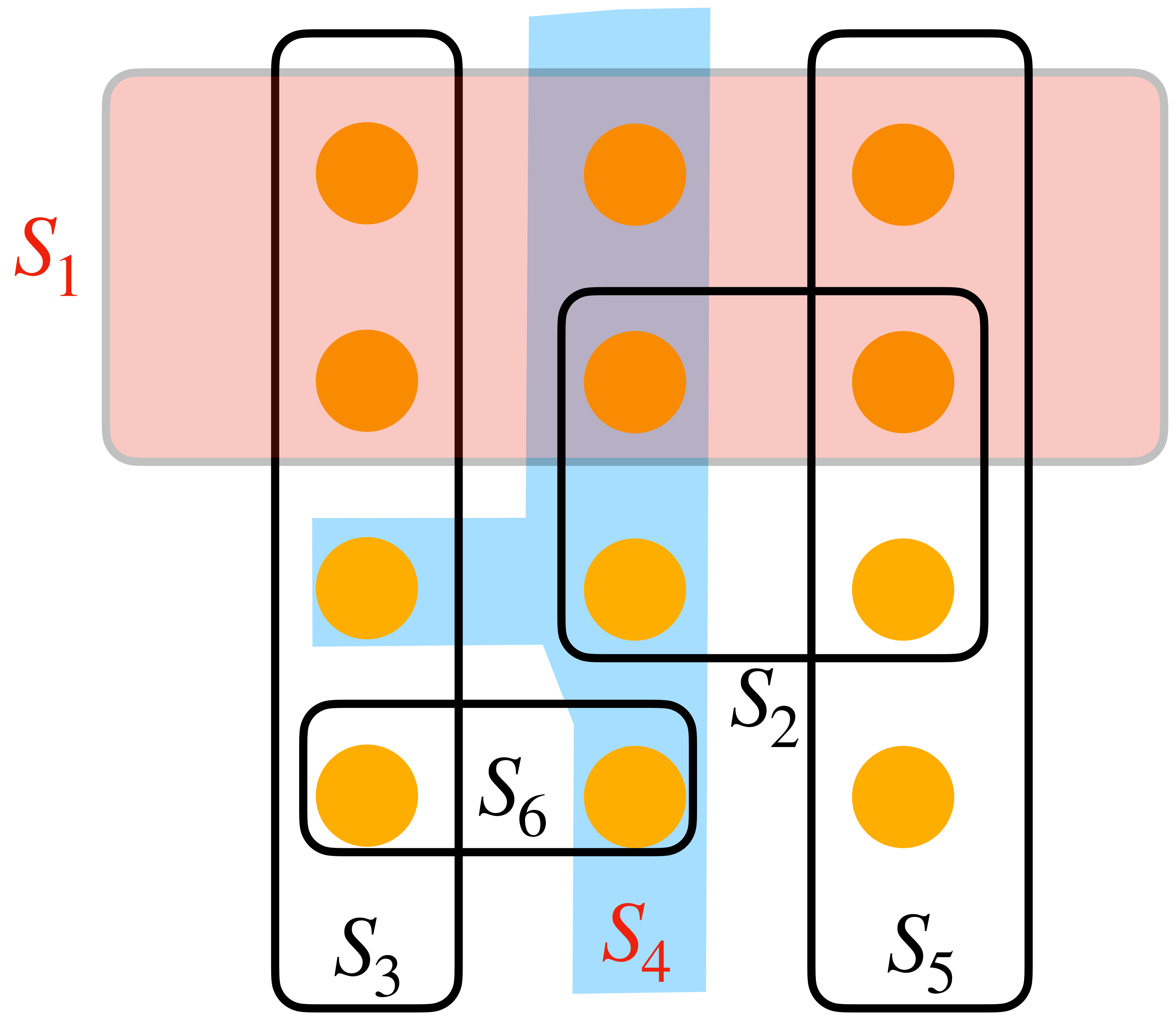
$C = \{S_3, S_4, S_5\}$

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

$U_0 = X$
$C = \varnothing$
$i = 0$
while $U_i \neq \varnothing$
    select $S \in F$ that maximizes $|S \cap U_i|$
    $U_{i+1} = U_i - S$
    $C = C \cup \{S\}$
    $i = i + 1$
return C

Idea: Each time, pick a subset that covers as many new elements as possible.



X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

$C = \{S_3, S_4, S_5\}$

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

$U_0 = X$

$C = \varnothing$

$i = 0$

while $U_i \neq \varnothing$

    select $S \in F$ that maximizes $|S \cap U_i|$

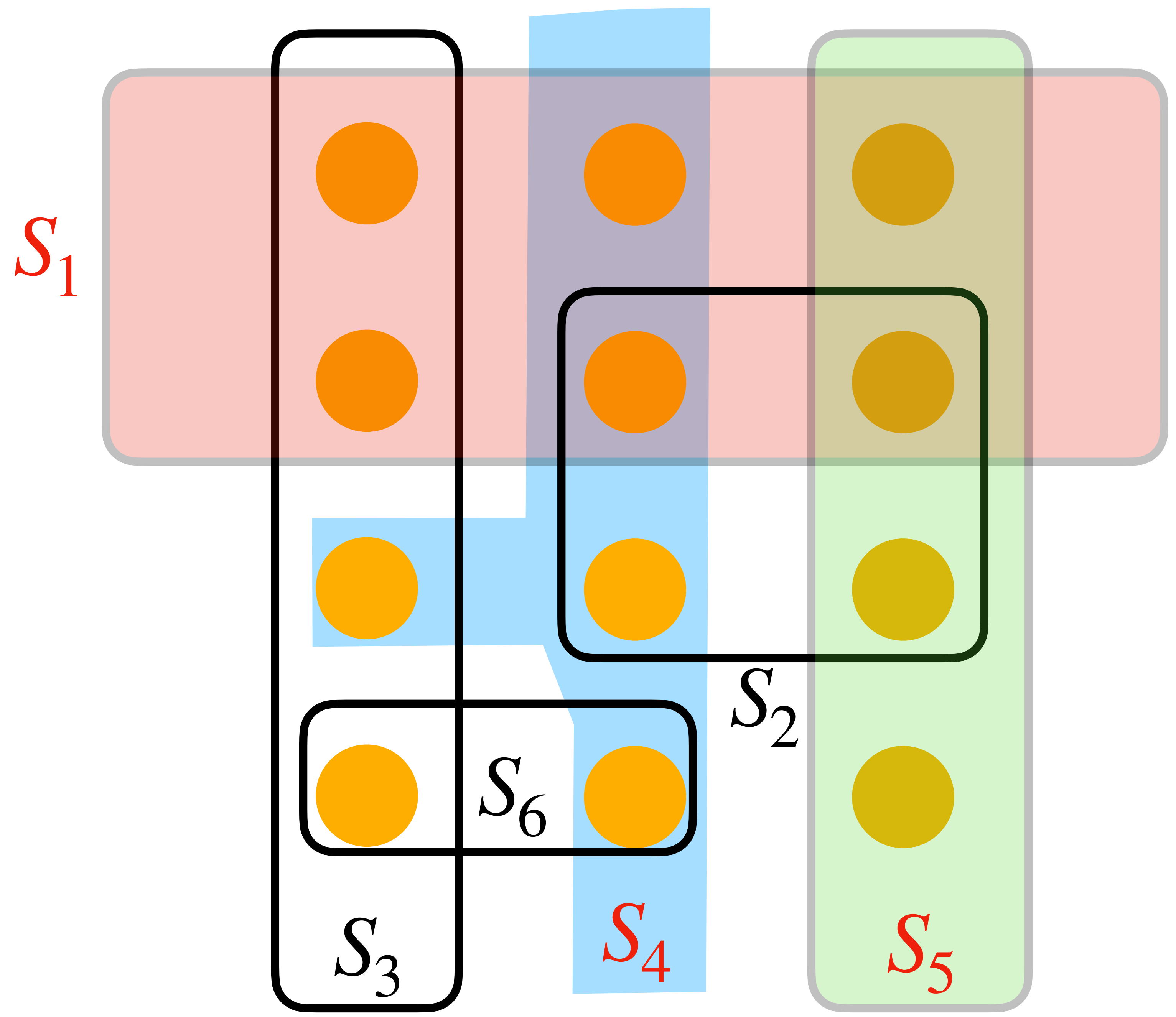
    $U_{i+1} = U_i - S$

    $C = C \cup \{S\}$

    $i = i + 1$

return C

Idea: Each time, pick a subset that covers as many new elements as possible.



X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

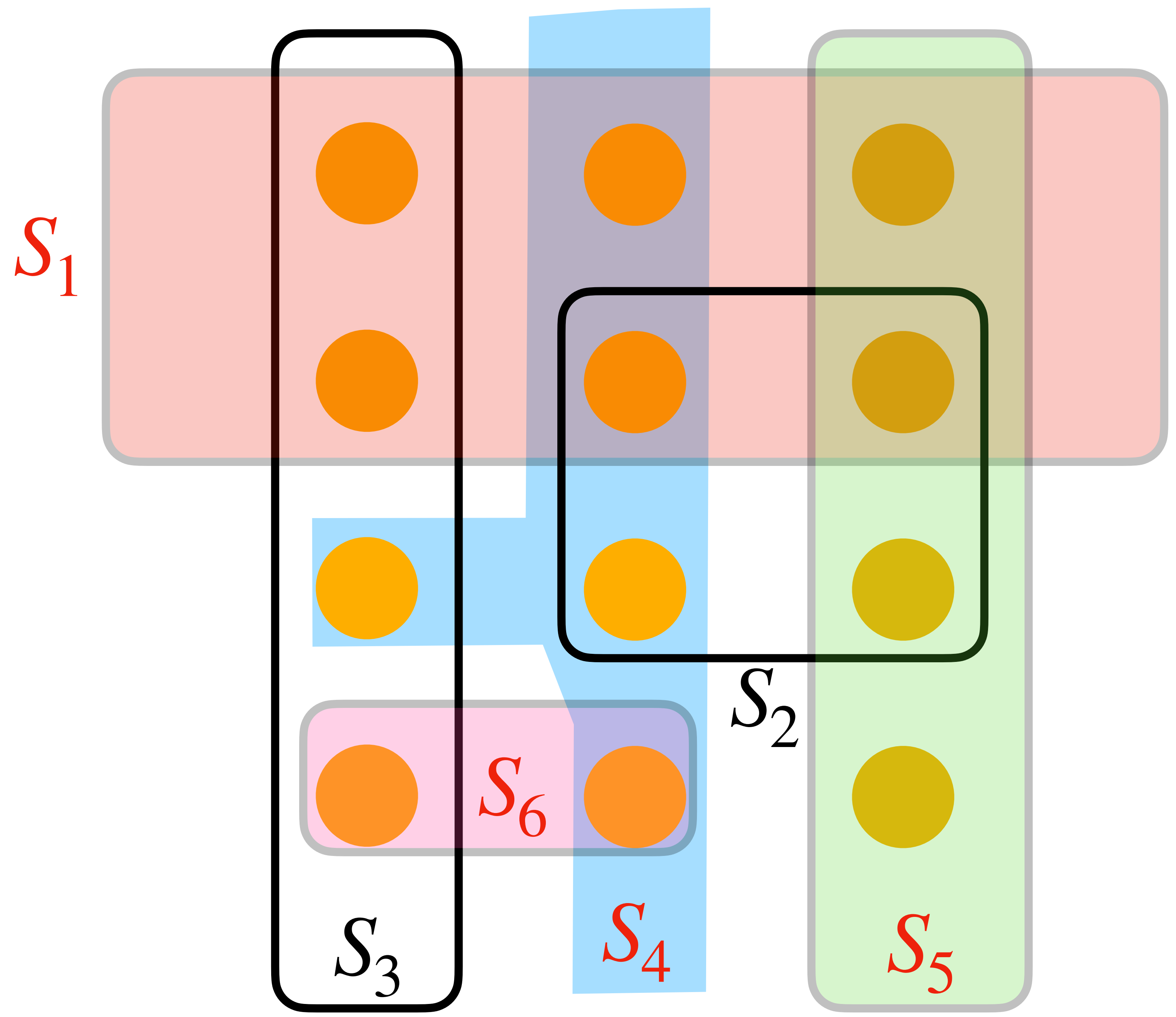
$C = \{S_3, S_4, S_5\}$

A Greedy Approximation Algorithm

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \emptyset$
3. $i = 0$
4. while $U_i \neq \emptyset$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

Idea: Each time, pick a subset that covers as many new elements as possible.



X : 12 elements

$F = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Optimal solution:

$C = \{S_3, S_4, S_5\}$

Quiz questions:

1. What is main idea of the above approximation algorithm for “Set Covering Problem”?

2. Can you think of an instance for which the above algorithm outputs an optimal solution, and an instance for which it does not?

Roadmap of this lecture:

1. Understand approximation algorithms by solving the "Set Covering Problem"
    1.1 Define "Set Covering Problem".
    1.2 A greedy approximation algorithm for "Set Covering Problem".
    1.3 Analyze the approximation ratio of the algorithm.

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Polynomial time.

Greedy-Set-Cover (X, F)

$U_0 = X$

$C = \emptyset$

$i = 0$

while $U_i \neq \emptyset$ At most $\min\{|X|, |F|\} = O(|X| + |F|)$ iterations

  select $S \in F$ that maximizes $|S \cap U_i|$

  $U_{i+1} = U_i - S$ In each iteration, time complexity is at most $O(|X| \cdot |F|)$

  $C = C \cup \{S\}$

  $i = i + 1$

return C

Time complexity of algorithm: $O(|X| \cdot |F| \cdot (|X| + |F|))$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

$U_0 = X$
$C = \varnothing$
$i = 0$
while $U_i \neq \varnothing$
    select $S \in F$ that maximizes $|S \cap U_i|$
    $U_{i+1} = U_i - S$
    $C = C \cup \{S\}$
    $i = i + 1$
return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

Claim: Every U_i can be covered by at most k^* subsets

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

Claim: Every U_i can be covered by at most k^* subsets

Claim: S covers at least $\dfrac{|U_i|}{k^*}$ elements in U_i

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

Claim: Every U_i can be covered by at most k^* subsets

Claim: S covers at least $\dfrac{|U_i|}{k^*}$ elements in U_i

$$|U_{i+1}| \leq |U_i| - \frac{|U_i|}{k^*} = |U_i|(1 - 1/k^*)$$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \emptyset$
3. $i = 0$
4. while $U_i \neq \emptyset$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

Claim: Every U_i can be covered by at most k^* subsets

Claim: S covers at least $\frac{|U_i|}{k^*}$ elements in U_i

$$|U_{i+1}| \leq |U_i| - \frac{|U_i|}{k^*} = |U_i|(1 - 1/k^*)$$

$$|U_0| = |X|$$

$$|U_1| \leq |U_0|(1 - 1/k^*) = |X|(1 - 1/k^*)$$

$$|U_2| \leq |U_1|(1 - 1/k^*) \leq |X|(1 - 1/k^*)^2$$

$$\ldots$$

$$|U_i| \leq |X|(1 - 1/k^*)^i$$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

$|U_i| \leq |X|(1 - 1/k^*)^i$

Let $c = \lceil \ln |X| \rceil$, then

$|U_{ck^*}| \leq |X|(1 - 1/k^*)^{ck^*} = |X|[(1 - 1/k^*)^{k^*}]^c$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

$|U_i| \leq |X|(1 - 1/k^*)^i$

Let $c = \lceil \ln |X| \rceil$, then

$|U_{ck^*}| \leq |X|(1 - 1/k^*)^{ck^*} = |X|[(1 - 1/k^*)^{k^*}]^c$

Known fact: $1 + x \leq e^x$ for all real x

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

$|U_i| \leq |X|(1 - 1/k^*)^i$

Let $c = \lceil \ln |X| \rceil$, then

$|U_{ck^*}| \leq |X|(1 - 1/k^*)^{ck^*} = |X|[(1 - 1/k^*)^{k^*}]^c$

Known fact: $1 + x \leq e^x$ for all real x

$$|U_{ck^*}| < |X|[(e^{-1/k^*})^{k^*}]^c = |X|e^{-c} = |X|e^{-\lceil \ln |X| \rceil}$$

$$\leq |X|e^{-\ln |X|} = |X| \cdot \frac{1}{|X|} = 1$$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

$|U_i| \leq |X|(1 - 1/k^*)^i$

Let $c = \lceil \ln |X| \rceil$, then

$|U_{ck^*}| \leq |X|(1 - 1/k^*)^{ck^*} = |X|[(1 - 1/k^*)^{k^*}]^c$

Known fact: $1 + x \leq e^x$ for all real x

$$|U_{ck^*}| < |X|[(e^{-1/k^*})^{k^*}]^c = |X|e^{-c} = |X|e^{-\lceil \ln|X| \rceil}$$

$$\leq |X|e^{-\ln|X|} = |X| \cdot \frac{1}{|X|} = 1$$

$k \leq ck^*$

Theorem: The algorithm is a polynomial-time $O(\lg X)$-approximation algorithm.

Proof: Let's analyze the approximation ratio.

Greedy-Set-Cover (X, F)

1. $U_0 = X$
2. $C = \varnothing$
3. $i = 0$
4. while $U_i \neq \varnothing$
5. select $S \in F$ that maximizes $|S \cap U_i|$
6. $U_{i+1} = U_i - S$
7. $C = C \cup \{S\}$
8. $i = i + 1$
9. return C

k^* : size of an optimal set cover

k : size of the set cover returned by the algorithm

$|U_i| \leq |X|(1 - 1/k^*)^i$

Let $c = \lceil \ln |X| \rceil$, then

$|U_{ck^*}| \leq |X|(1 - 1/k^*)^{ck^*} = |X|[(1 - 1/k^*)^{k^*}]^c$

Known fact: $1 + x \leq e^x$ for all real x

$$|U_{ck^*}| < |X|[(e^{-1/k^*})^{k^*}]^c = |X|e^{-c} = |X|e^{-\lceil \ln|X| \rceil}$$

$$\leq |X|e^{-\ln|X|} = |X| \cdot \frac{1}{|X|} = 1$$

$k \leq ck^*$ $\Rightarrow$ $\frac{k}{k^*} \leq c = \lceil \ln |X| \rceil$ $O(\ln X)$

Quiz questions:

1. What is the main method we used to find the approximation ratio of the algorithm for "Set Covering Problem"?

2. Is the above approximation ratio a constant, or a function that grows with the input size of the problem?