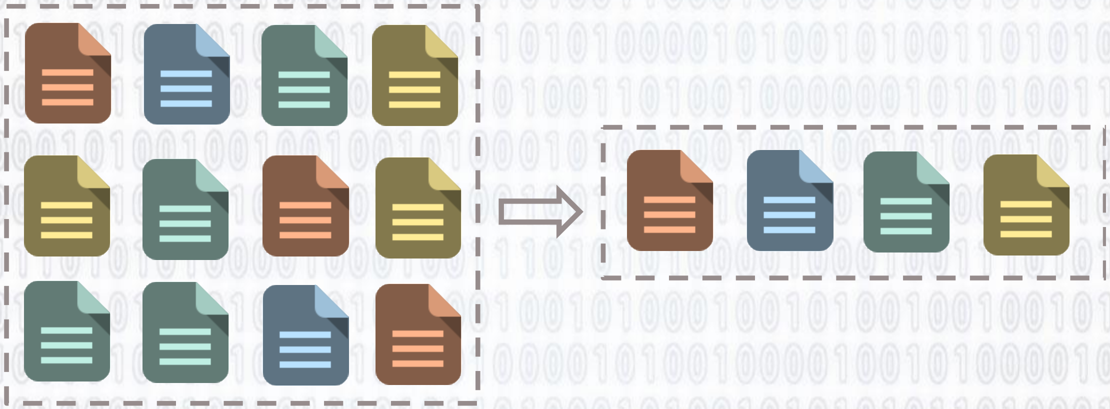
Can coding reduce fragmentation in deduplicated storage systems?



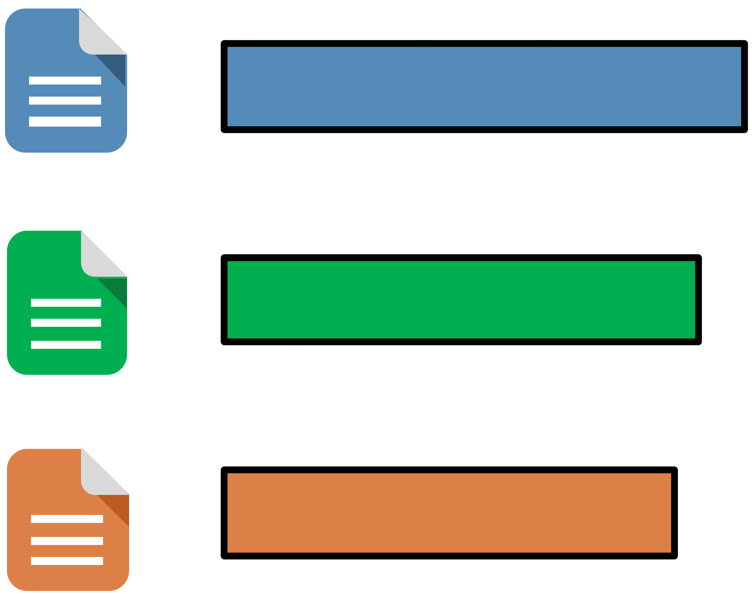
Yun-Han Li, Jin Sima, Ilan Shomorony and Olgica Milenkovic

UIUC ECE

ITA 2/14/2025

Data Deduplication

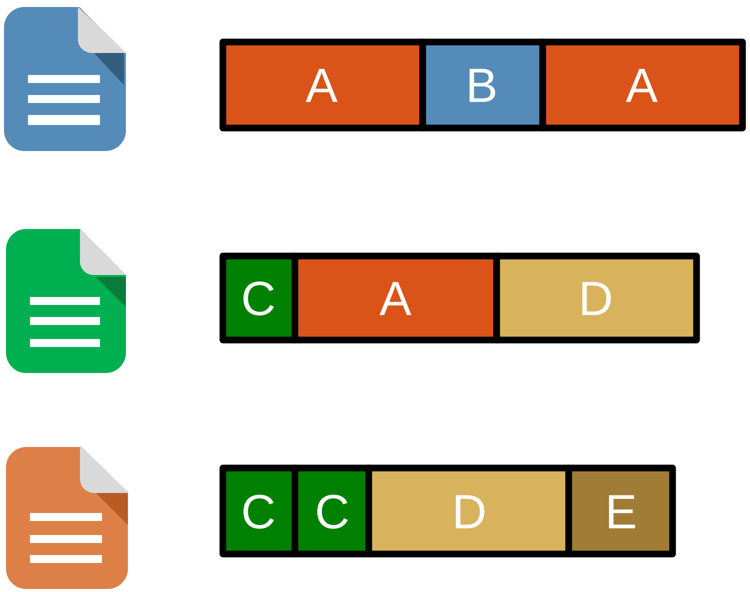
- Identify and remove duplicated “chunks” in data storage
- Used in data centers and cloud storage



- Two steps:
  1. Chunking

Data Deduplication

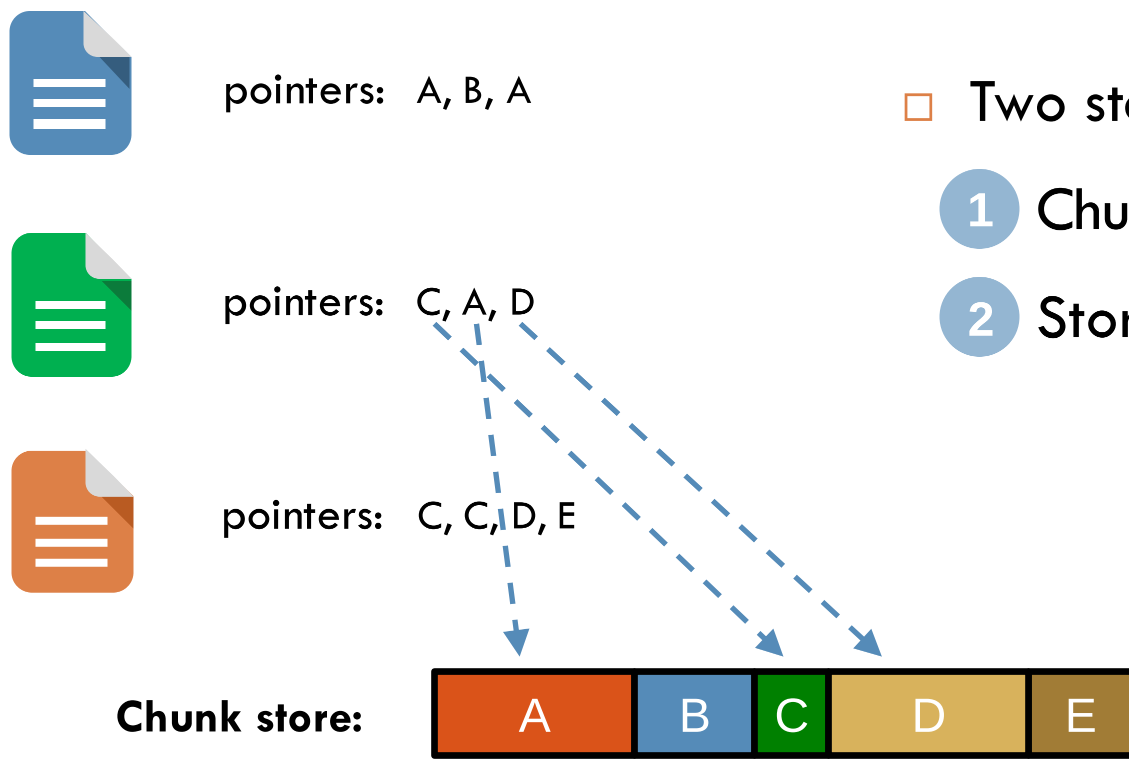
- Identify and remove duplicated "chunks" in data storage
- Used in data centers and cloud storage



- Two steps:
  1. Chunking

Data Deduplication

- Identify and remove duplicated "chunks" in data storage
- Used in data centers and cloud storage



- Two steps:
  1. Chunking
  2. Storing chunks

Data Deduplication

- Identify and remove duplicated “chunks” in data storage
- Used in data centers and cloud storage

pointers: A, B, A

pointers: C, A, D

pointers: C, C, D, E

- Two steps:
  1. Chunking
  2. Storing chunks

Chunk store: | A | B | C | D | E |

Urs Niesen, “An Information-Theoretic Analysis of Deduplication”, 2017
Lou, Farnoud, “Data Deduplication with Random Substitutions”, 2022

Data Deduplication

- Identify and remove duplicated “chunks” in data storage
- Used in data centers and cloud storage

pointers: A, B, A

pointers: C, A, D

pointers: C, C, D, E

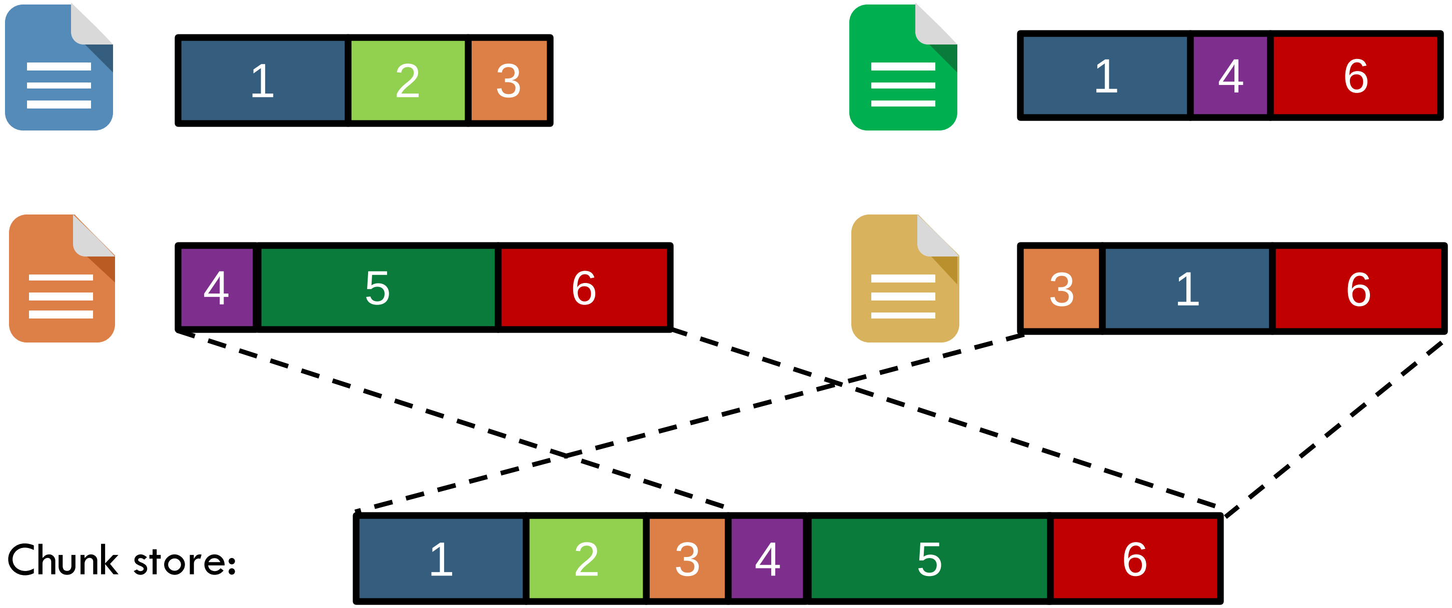
- Two steps:
  1. Chunking
  2. Storing chunks

Chunk store:

A	B	C	D	E

Issue: data fragmentation

- Files are fragmented across the chunk store

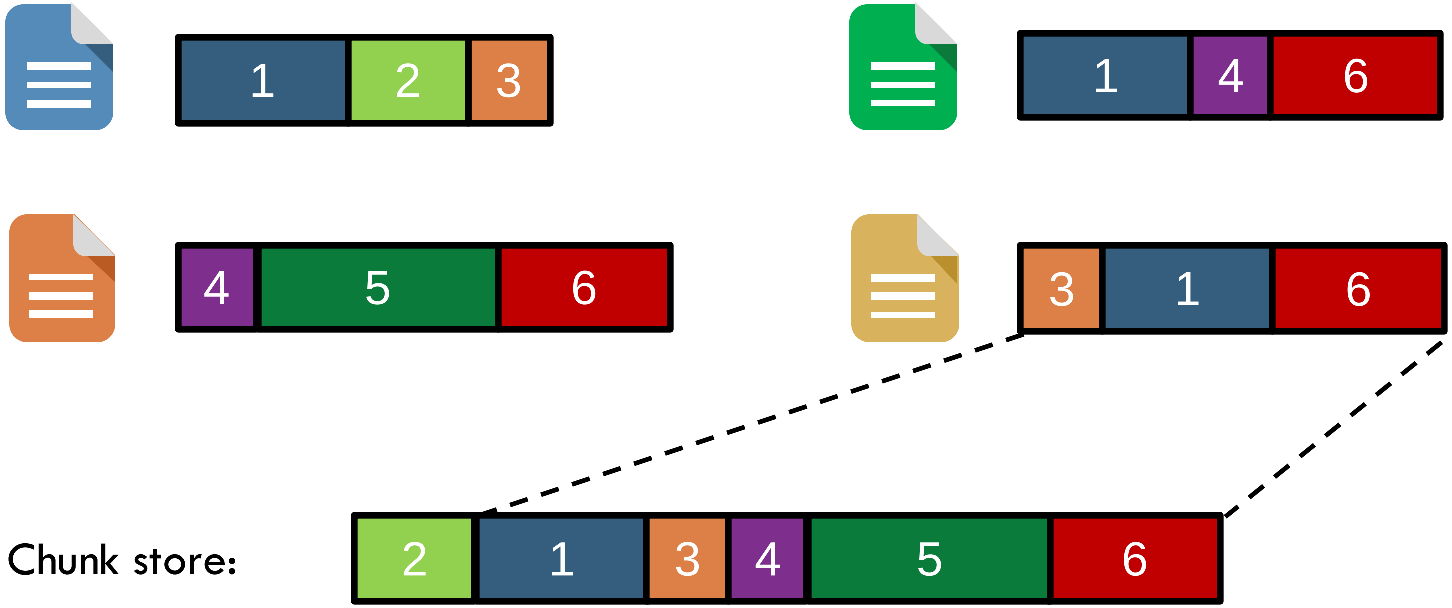


Need to read longer stretches of memory (or make jumps)

- How do we optimally arrange the chunk store?

Issue: data fragmentation

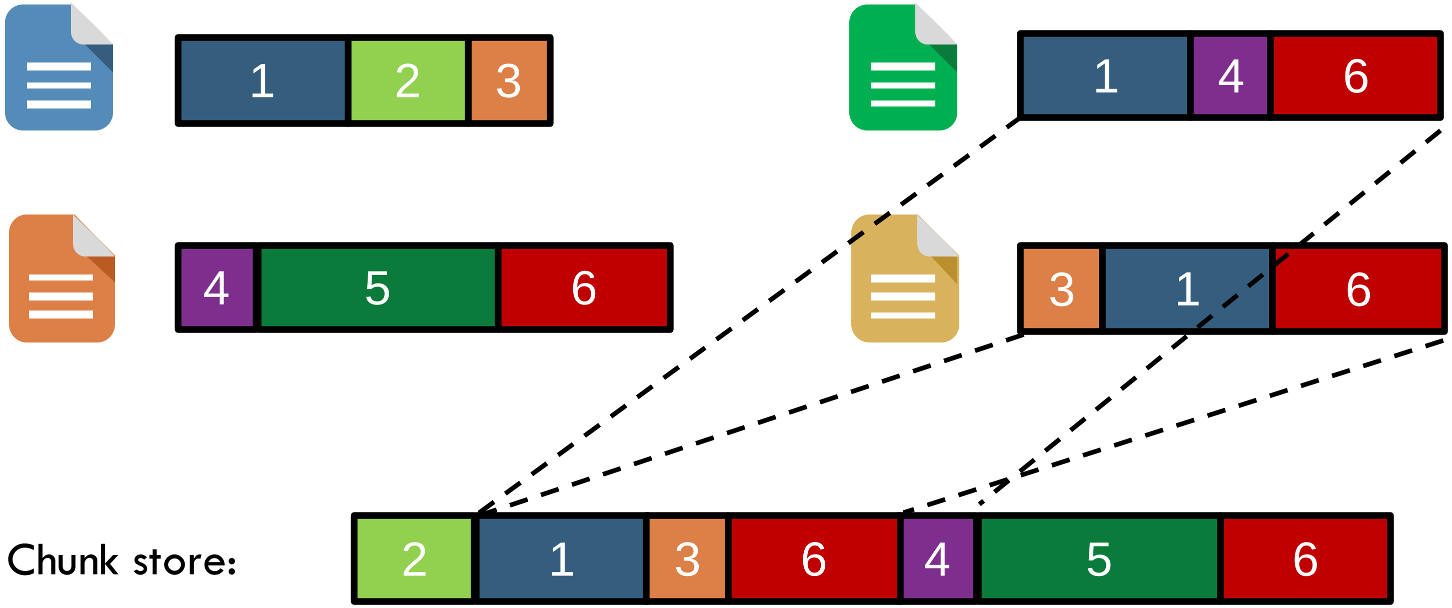
- Files are fragmented across the chunk store



- How do we optimally arrange the chunk store?

Issue: data fragmentation

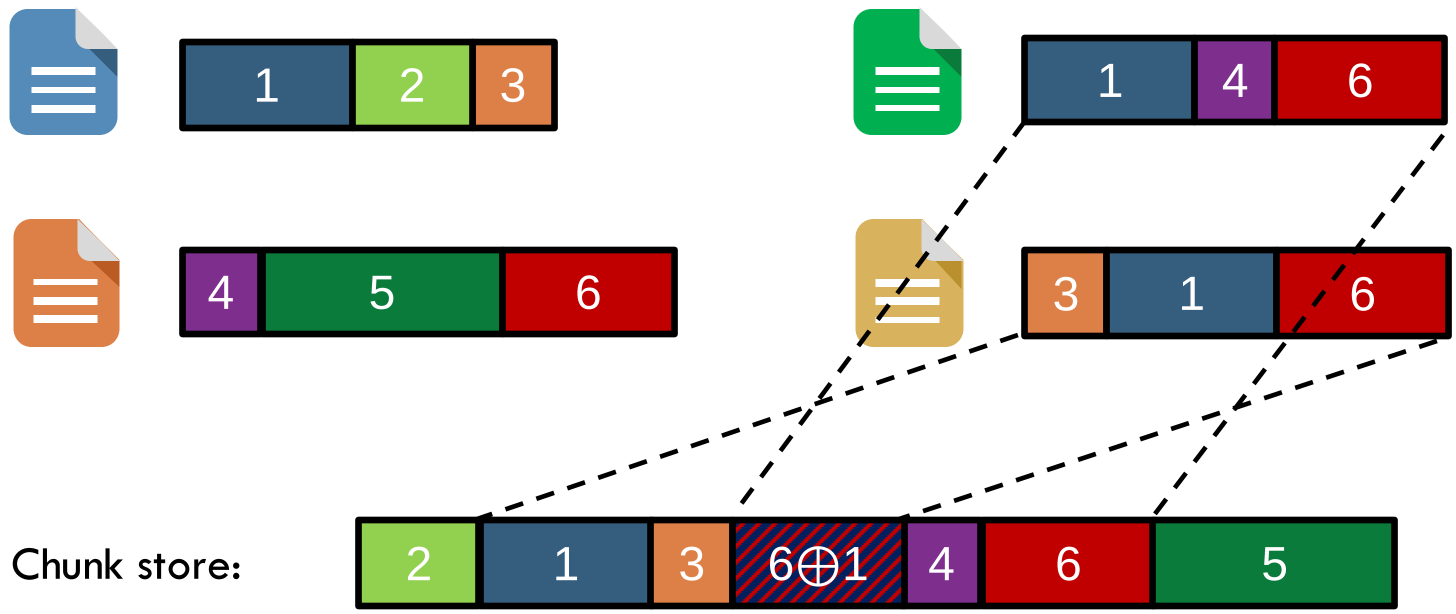
- Files are fragmented across the chunk store



- How do we optimally arrange the chunk store?
- Can we reduce fragmentation by adding redundancy back?

Issue: data fragmentation

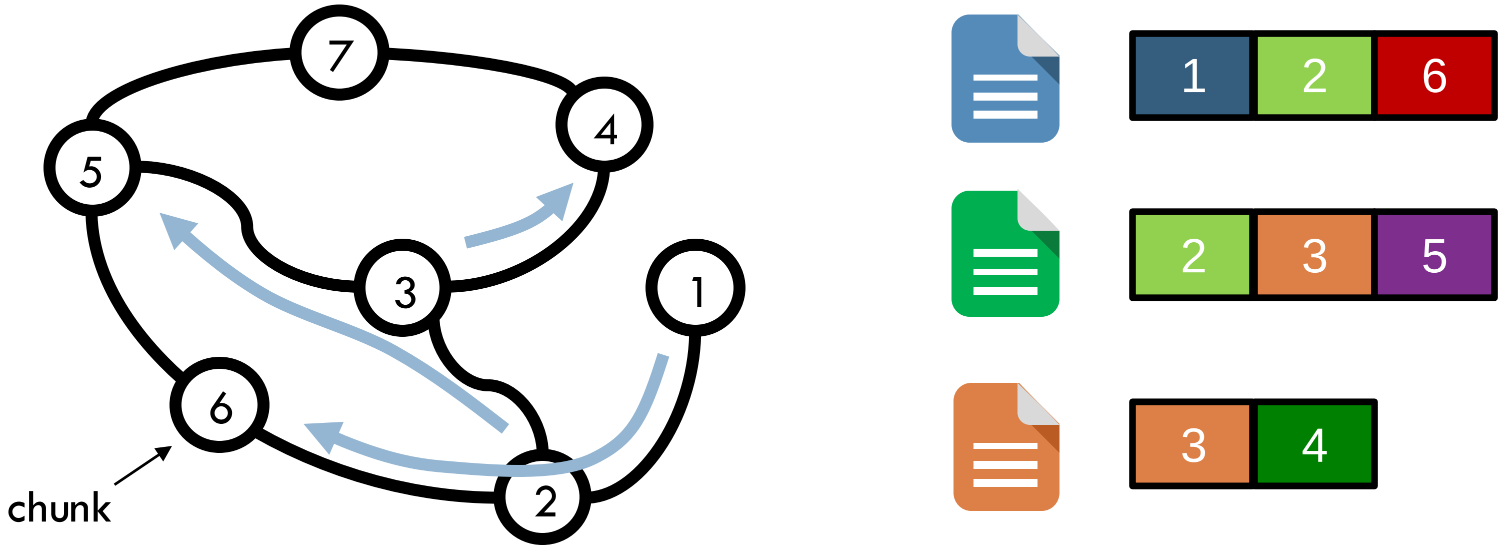
- Files are fragmented across the chunk store



- How do we optimally arrange the chunk store?
- Can we reduce fragmentation by adding redundancy back?
- Can coding help?

File model

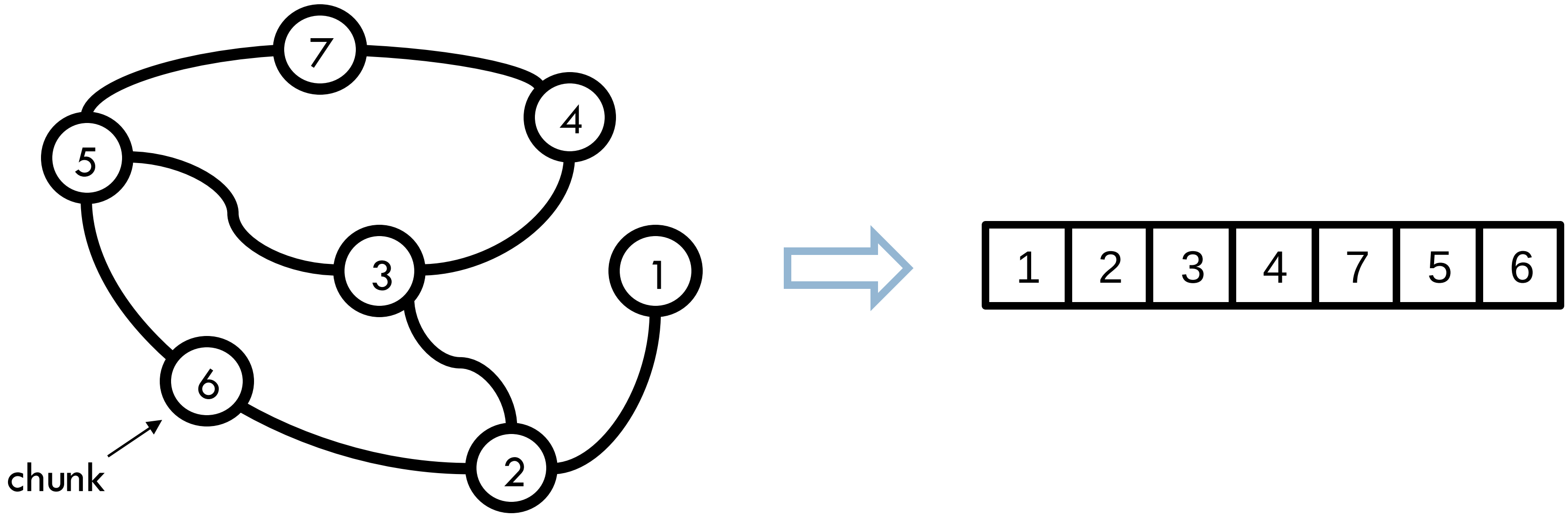
- Undirected graph $G(V, E)$ where chunks are nodes
- Files are paths on the graph



- Set of files: $F_t = \{\text{all paths of length} \leq t\}$
- **Goal:** Find a "good" linear storage for all chunks

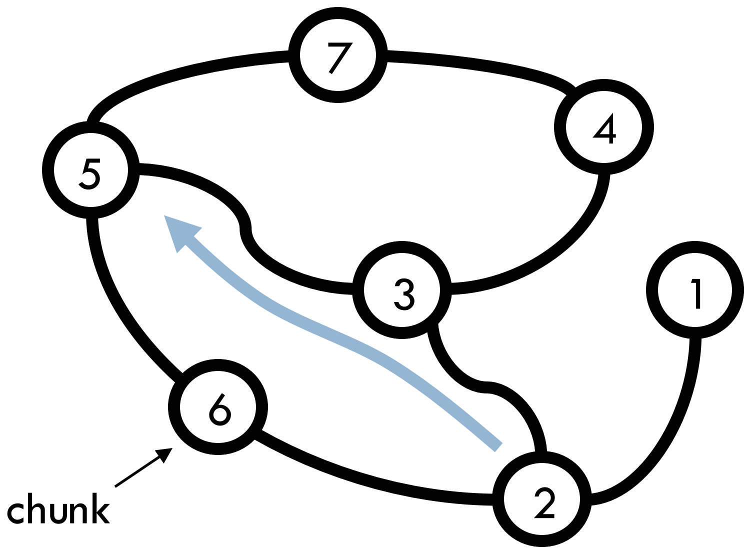
File model

- Undirected graph $G(V, E)$ where chunks are nodes
- Files are paths on the graph



- Set of files: $F_t = \{\text{all paths of length} \leq t\}$
- **Goal:** Find a “good” linear storage for all chunks

New metric for fragmentation



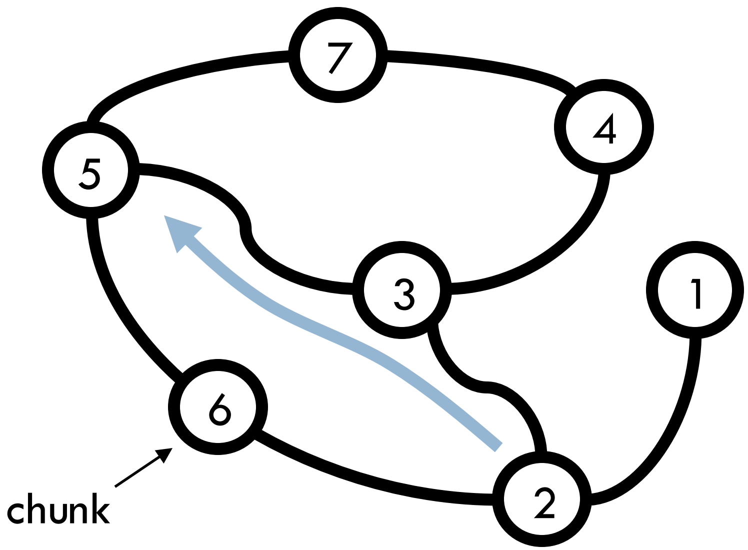
$$x = \boxed{1 \mid 2 \mid 3 \mid 4 \mid 7 \mid 5 \mid 6}$$

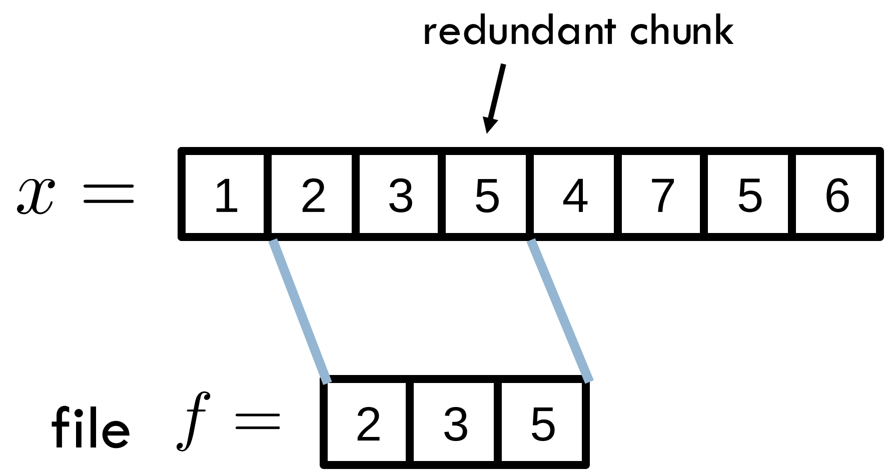
file $f = \boxed{2 \mid 3 \mid 5}$

$$\mathrm{stretch}(f, x) = 1$$

- Stretch of a file graph G: $S = \min_x \max_f \ \mathrm{stretch}(f, x)$

New metric for fragmentation



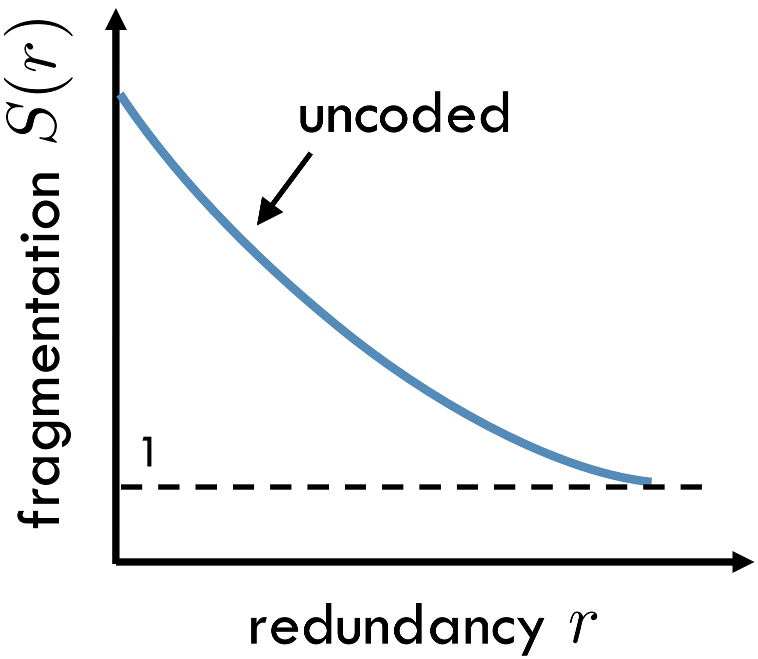


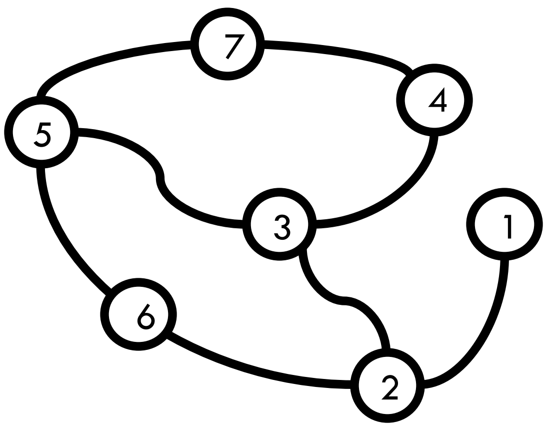
$$\mathrm{stretch}(f, x) = 1$$

- Stretch of a file graph G: $S = \min_x \max_f \ \mathrm{stretch}(f, x)$
- Using r redundant chunks: $S(r) = \min_x \max_f \ \mathrm{stretch}(f, x)$

can have r redundant chunks

Redundancy-fragmentation tradeoff





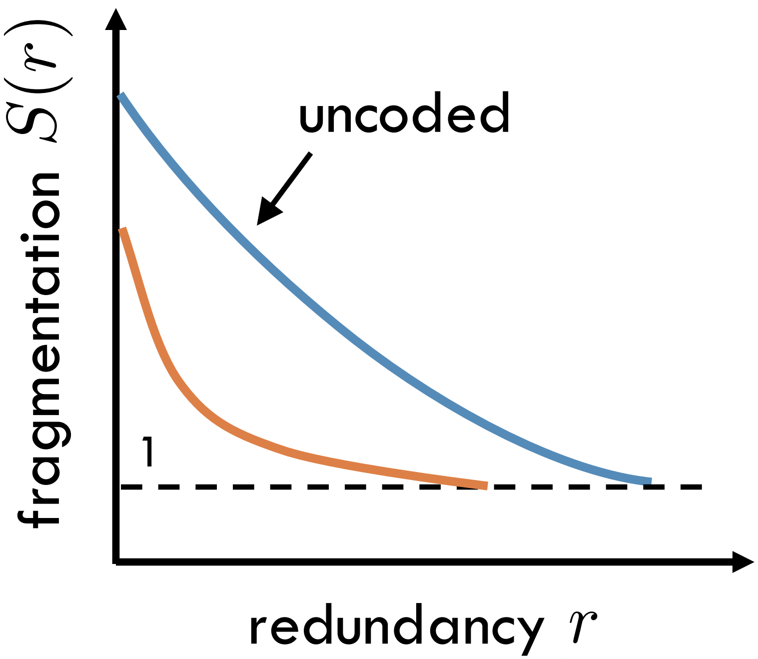
$x =$

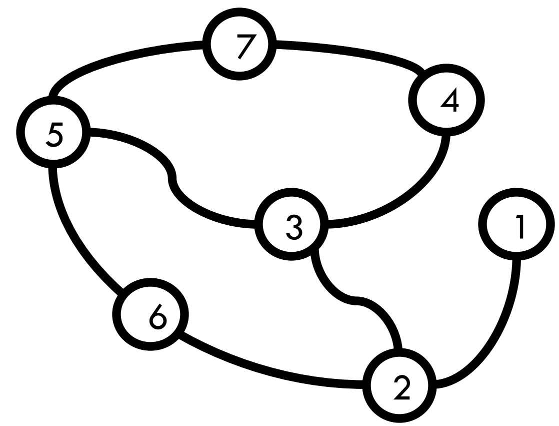
1	2	3	5	4	7	5	6

- Using r redundant chunks: $$S(r) = \min_x \max_f \; \mathrm{stretch}(f, x)$$

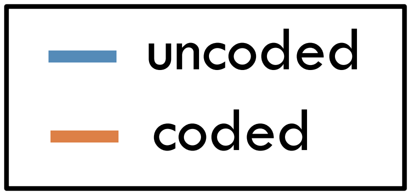
can have r redundant chunks

Redundancy-fragmentation tradeoff



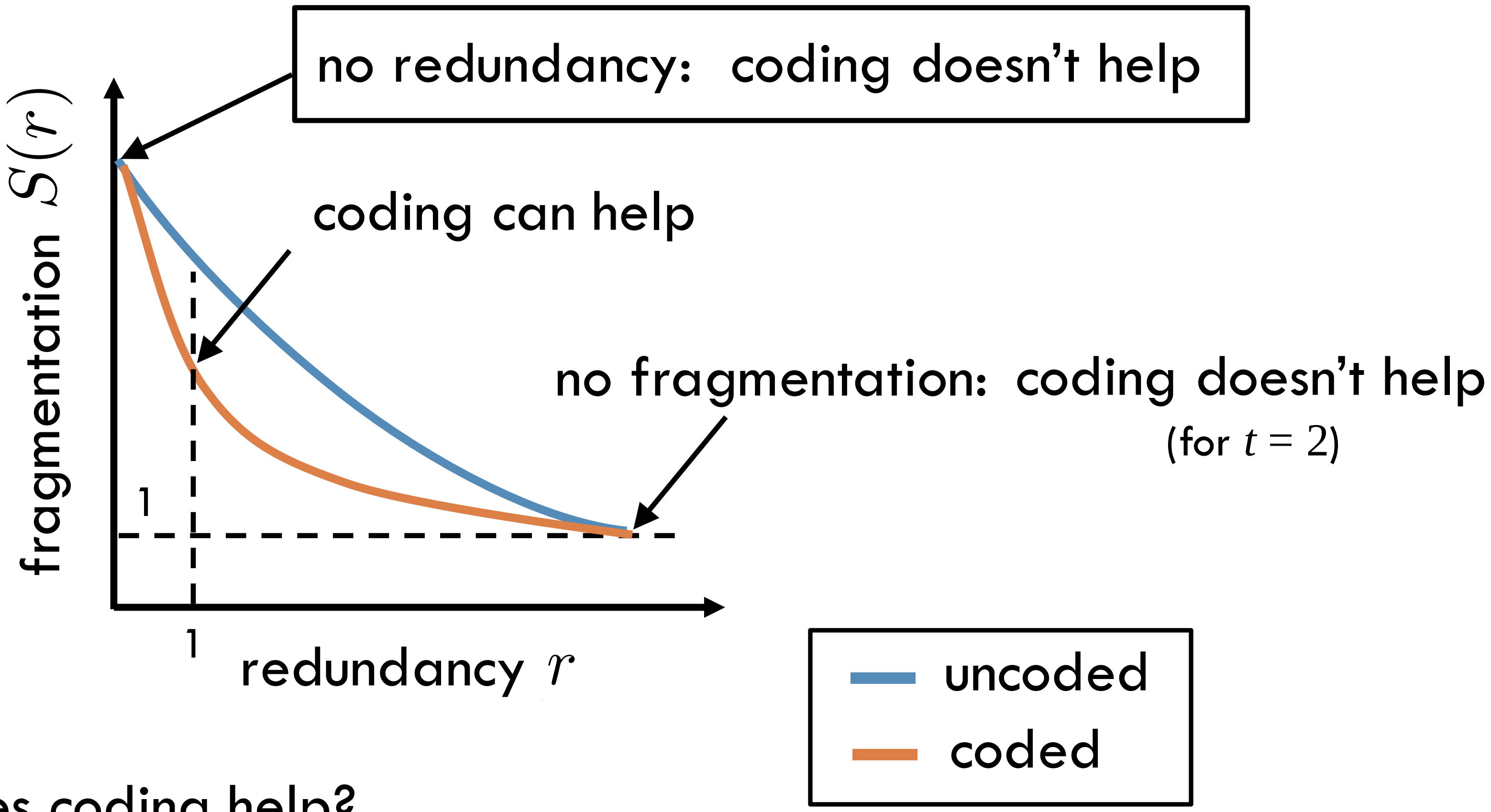


$x =$ | 1 | 2 | 3 | c_1 | 4 | 7 | c_2 | 6 |



- Does coding help?

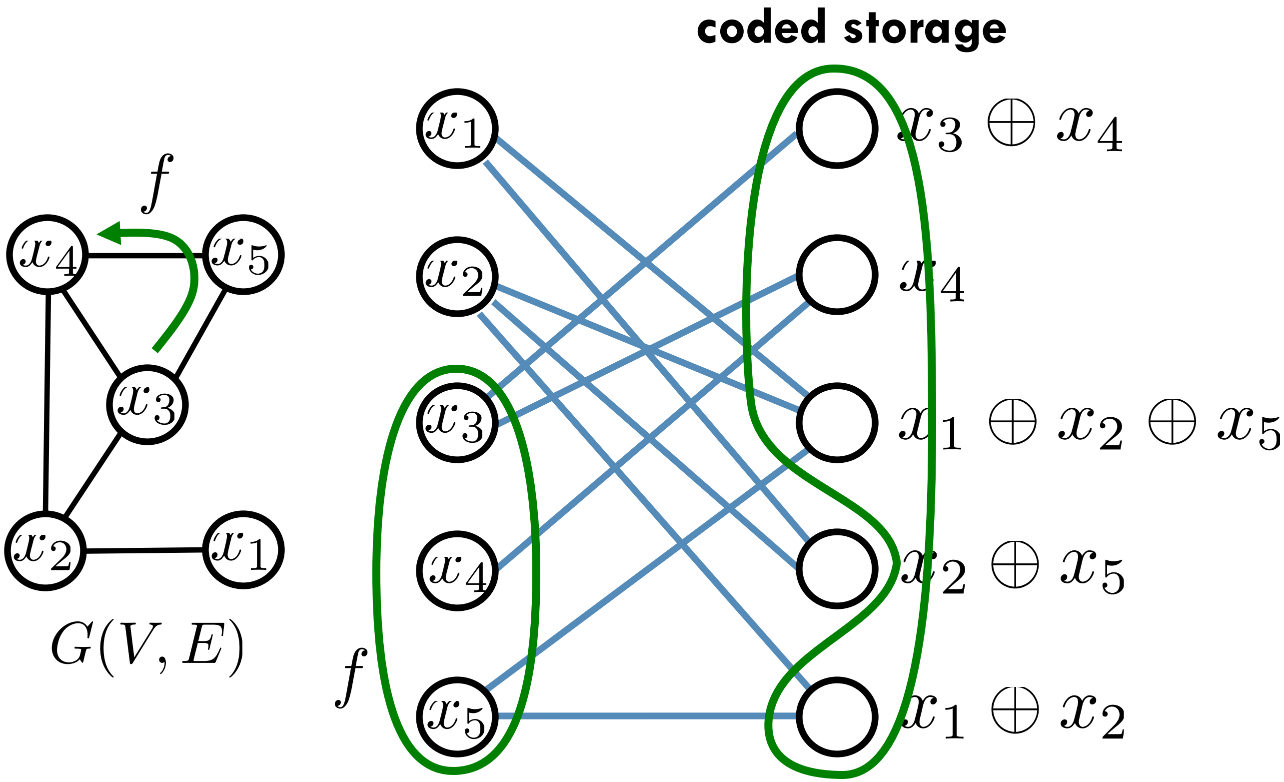
Redundancy-fragmentation tradeoff



- Does coding help?
- We study several special cases [1]

[1] Li, Sima, **S.**, Milenkovic, "Reducing Fragmentation in Data Deduplication Systems via Partial Repetition and Coding"

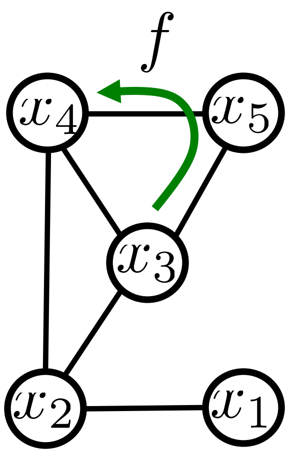
No redundancy: coding doesn't help



$$|\mathrm{Rec}(f)| \geq |f|$$

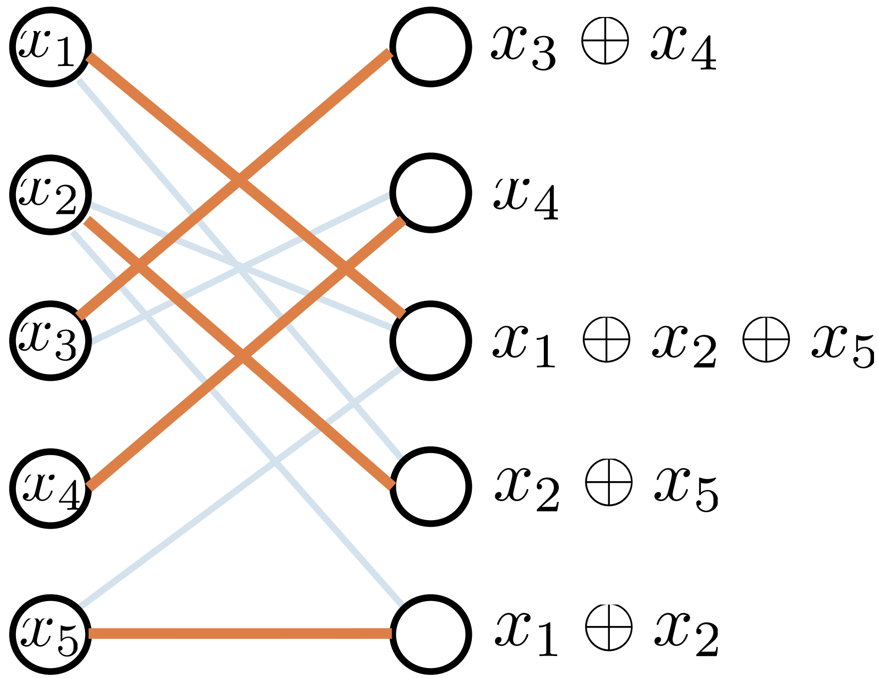
- **Lemma:** Every chunk has a unique minimal recovery set
- Hall's marriage condition $\implies$ perfect matching exists

No redundancy: coding doesn't help



$G(V, E)$

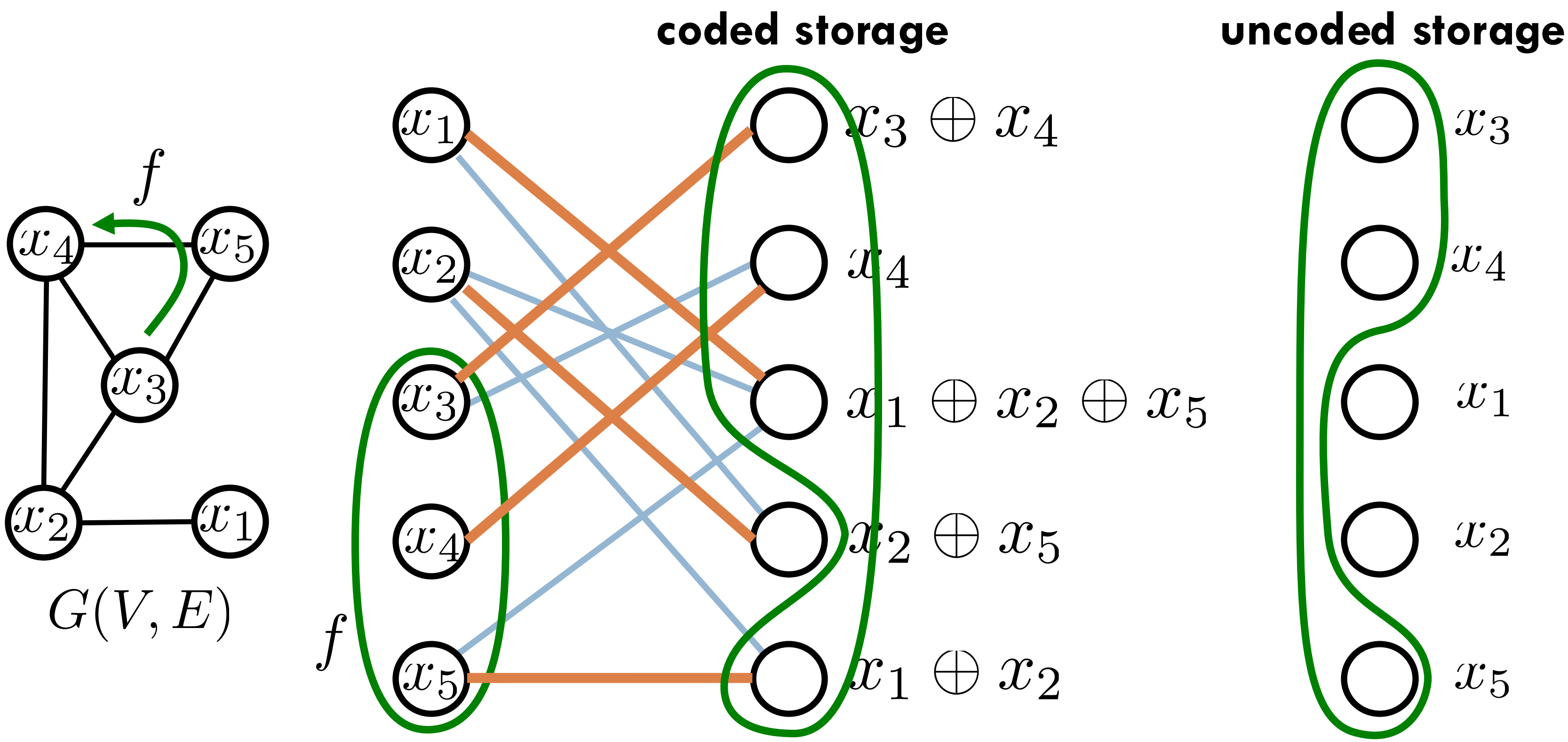
coded storage



$$|\mathrm{Rec}(f)| \geq |f|$$

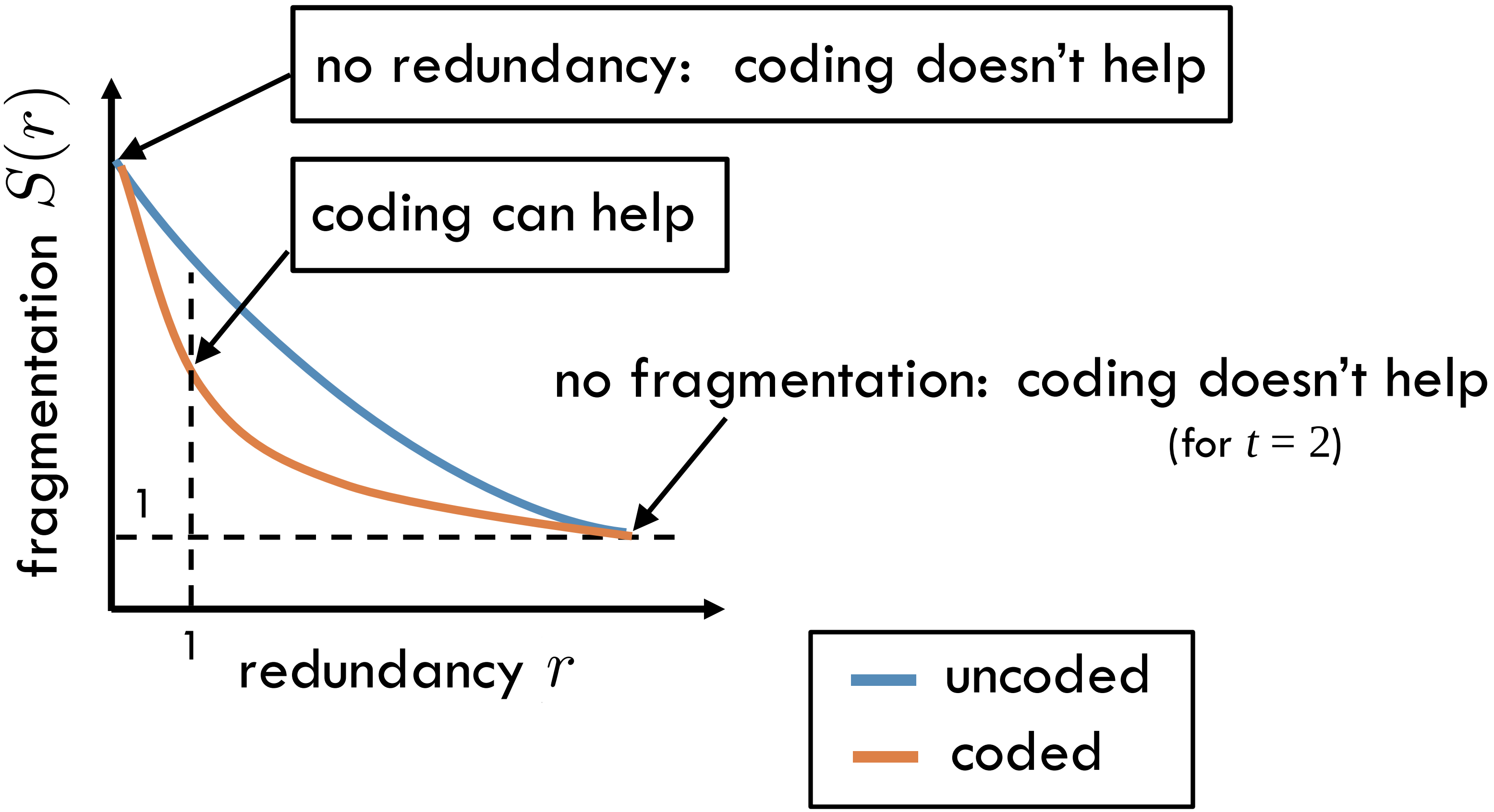
- **Lemma:** Every chunk has a unique minimal recovery set
- Hall's marriage condition $\Longrightarrow$ perfect matching exists

No redundancy: coding doesn't help



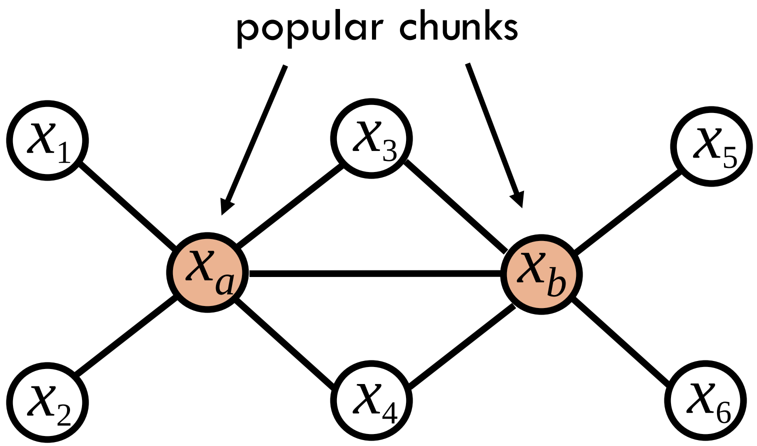
- **Lemma:** Every chunk has a unique minimal recovery set
- Hall's marriage condition $\Longrightarrow$ perfect matching exists
- Fragmentation can only improve!

Redundancy-fragmentation tradeoff



Coding with one redundant chunk

$G(V, E)$



Set of files:

$F_2 = \{(x_i, x_j) \in E\}$

coded storage: $x_1 \quad x_2 \quad \underbrace{\mathbf{x_a} \quad x_3 \quad \overbrace{\mathbf{x_a} \oplus \mathbf{x_b}}}_{(x_3,\, x_b)}$ ... $\overbrace{\mathbf{x_a} \oplus \mathbf{x_b} \quad x_4 \quad \mathbf{x_b}}^{(x_4,\, x_a)} \quad x_5 \quad x_6$

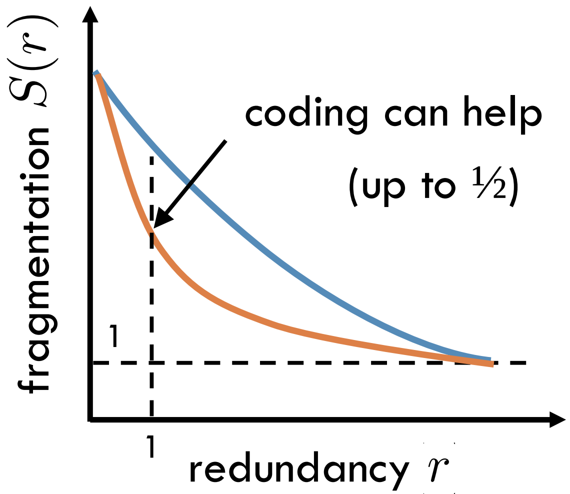
worst-case stretch = 3/2

uncoded storage: $x_1 \quad x_2 \quad \mathbf{x_a} \quad \underbrace{x_3 \quad \mathbf{x_a} \quad x_4 \quad \mathbf{x_b}}_{(x_3,\, x_b)} \quad x_5 \quad x_6$

worst-case stretch = 2

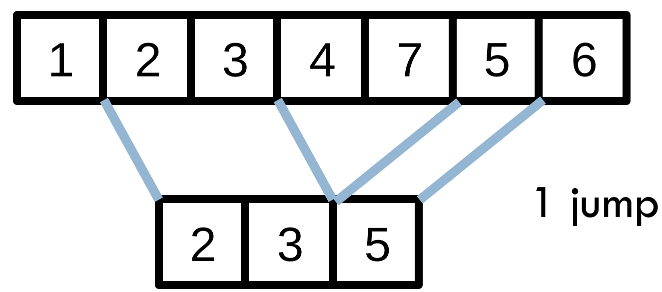
Coding for deduplication

- Worst-case stretch metric $S(r)$
  - Coding can help, but not too much
  - Computing $S(r)$ is NP-hard in general
  - Special classes of file graphs [1]



- Probabilistic file model? [2,3]
  - Average-case stretch

- Different fragmentation metrics?
  - Jump metric [1]



Thank you!

[1] Li, Sima, **S.**, Milenkovic, "Reducing Fragmentation in Data Deduplication Systems via Partial Repetition and Coding"
[2] Coffey, Klimesh, "Fundamental limits for information retrieval", 2003
[3] Niesen, "An Information-Theoretic Analysis of Deduplication", 2017