# Fundamental Limits of Coded Caching: From Uncoded Prefetching to Coded Prefetching

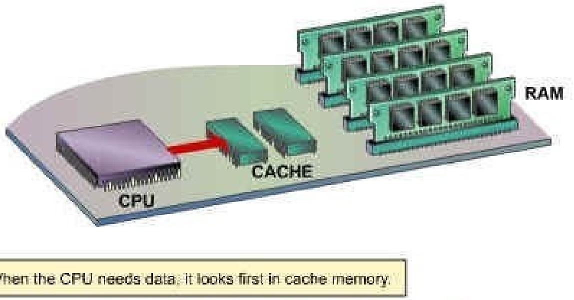
Kai Zhang and Chao Tian

Texas A&M University

ISIT 2018

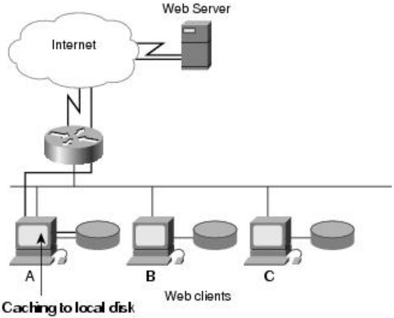
# Coded Caching and Its Applications

- A data management strategy to reduce delay during peak-traffic time;
- Single user setting, e.g. on-CPU caches, RAM in computers;
- Multiple user setting, e.g. networked system.



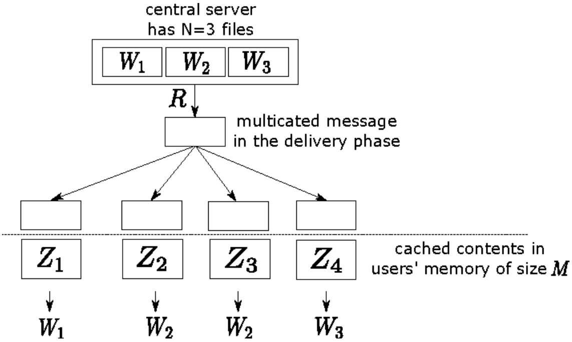
# Coded Caching and Its Applications

- A data management strategy to reduce delay during peak-traffic time;
- Single user setting, e.g. on-CPU caches, RAM in computers;
- Multiple user setting, e.g. networked system.



# An Information-theoretic Formulation

- $N$ files of same size, $K$ users, each with local cache of size $M$;
- Prefetching phase: users fill cache before knowing requests;
- Delivery phase: each user requests a single file, server multicasts information to accommodate requests.



# Existing Schemes Using Uncoded Prefetching

Maddah-Ali & Niesen[1] scheme uses uncoded prefetching,

- Prefetching strategy:
  - Partition each file into $\binom{K}{t}$ segments, each associated with a cardinality-$t$ subset;
  - Each user caches $\binom{K-1}{t-1}$ segments, e.g.: $(N, K) = (3, 4), t = 2$,

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
| User 2 | $A_{12}$ | $A_{23}$ | $A_{24}$ | $B_{12}$ | $B_{23}$ | $B_{24}$ | $C_{12}$ | $C_{23}$ | $C_{24}$ |
| User 3 | $A_{13}$ | $A_{23}$ | $A_{34}$ | $B_{13}$ | $B_{23}$ | $B_{34}$ | $C_{13}$ | $C_{23}$ | $C_{34}$ |
| User 4 | $A_{14}$ | $A_{24}$ | $A_{34}$ | $B_{14}$ | $B_{24}$ | $B_{34}$ | $C_{14}$ | $C_{24}$ | $C_{34}$ |

[1] Maddah-Ali M A, Niesen U. Fundamental limits of caching[J]. IEEE Transactions on Information Theory, 2014, 60(5): 2856-2867.

# Existing Schemes Using Uncoded Prefetching

- Delivery strategy
  - For any $(t+1)$-user set $\mathcal{B}$, send *XOR* of requested segments, e.g., demand (A,A,B,C), server transmits:

$$
\begin{aligned}
\mathcal{B} = \{1,2,3\} &: A_{23} + A_{13} + B_{12};\\
\mathcal{B} = \{1,2,4\} &: A_{24} + A_{14} + C_{12};\\
\mathcal{B} = \{1,3,4\} &: A_{34} + B_{14} + C_{13};\\
\mathcal{B} = \{2,3,4\} &: A_{34} + B_{24} + C_{23}.
\end{aligned}
$$

An improved scheme by Yu et al[2] by identifying redundancies.

[2] Yu Q, Maddah-Ali M A, Avestimehr A S. The exact rate-memory tradeoff for caching with uncoded prefetching[J]. IEEE Transactions on Information Theory, 2017.

# Existing Schemes Using Uncoded Prefetching

- For example, user 1 decodes as:

| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
|---|---|---|---|---|---|---|---|---|---|

  - Step 1: using $A_{13}$ and $B_{12}$ to decode $A_{23}$;

$$A_{23} + A_{13} + B_{12}, \quad A_{24} + A_{14} + C_{12}$$
$$A_{34} + B_{14} + C_{13}, \quad A_{34} + B_{24} + C_{23}$$

# Existing Schemes Using Uncoded Prefetching

- For example, user 1 decodes as:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |

  - Step 2: using $A_{14}$ and $C_{12}$ to decode $A_{24}$;

$$A_{23} + A_{13} + B_{12}, \quad A_{24} + A_{14} + C_{12}$$
$$A_{34} + B_{14} + C_{13}, \quad A_{34} + B_{24} + C_{23}$$

# Existing Schemes Using Uncoded Prefetching

- For example, user 1 decodes as:

| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
|---|---|---|---|---|---|---|---|---|---|

  - Step 3: using $B_{14}$ and $C_{13}$ to decode $A_{34}$.

$$\begin{aligned} &A_{23} + A_{13} + B_{12}, \quad && A_{24} + A_{14} + C_{12} \\ &A_{34} + B_{14} + C_{13}, \quad && A_{34} + B_{24} + C_{23} \end{aligned}$$

# Existing Schemes Using Coded Prefetching

Tian-Chen scheme[1] uses coded prefetching,

- Prefetching strategy:
  - Partition each file into $\binom{K}{t}$ segments, each user allocated $\binom{K-1}{t-1}$ segments;
  - Encode using rank metric code, store the parities;
    e.g.: $(N, K) = (3, 4)$, $t = 2$, user 1 stores

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}_{(5\times 9)} \cdot$$

$$(A_{12}, A_{13}, A_{14}, B_{12}, B_{13}, B_{14}, C_{12}, C_{13}, C_{14})^T.$$

[1] Tian C, Chen J. Caching and delivery via interference elimination[C]. Information Theory (ISIT), 2016 IEEE International Symposium on. IEEE, 2016: 830-834.

# Existing Schemes Using Coded Prefetching

- Delivery strategy:
  - Linear combination of segments from one file;
  - Serve two roles: content delivery and resolve coded symbols;
  - E.g.: demand (A,A,B,C), transmits:

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13}+A_{23},\ A_{14}+A_{24}.$$

# Existing Schemes Using Coded Prefetching

- For example, user 1 decodes as:

$$
\begin{pmatrix}
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
 & & & \color{blue}{\times} & & & & & \\
 & & & & & \color{blue}{\times} & & & \\
 & & & & & & \color{blue}{\times} & & \\
 & & & & & & & \color{blue}{\times} &
\end{pmatrix}_{(5\times 9)} \cdot
$$

$$
(A_{12},\ A_{13},\ A_{14},\ B_{12},\ B_{13},\ B_{14},\ C_{12},\ C_{13},\ C_{14})^T
$$

  - Step 1: user 1 collects 4 symbols $\color{blue}{B_{12}, B_{14}, C_{12}, C_{13}}$, decodes all 9 symbols;

# Existing Schemes Using Coded Prefetching

- For example, user 1 decodes as:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |

  - Step 2: user 1 uses $A_{13}, A_{14}$ to decode $A_{23}, A_{24}$;

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13}+A_{23},\ A_{14}+A_{24}$$

# Existing Schemes Using Coded Prefetching

- For example, user 1 decodes as:

| User 1 | $A_{12}$ | $A_{13}$ | $A_{14}$ | $B_{12}$ | $B_{13}$ | $B_{14}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
|---|---|---|---|---|---|---|---|---|---|

  - Step 3: user 1 collects $A_{34}$.

$$A_{34},\ B_{12},\ B_{14},\ B_{24},\ C_{12},\ C_{13},\ C_{23},\ A_{13}+A_{23},\ A_{14}+A_{24}$$

# Two Quite Different Schemes

- Uncoded *vs.* coded prefetching;
- Binary *vs.* non-binary code;
- En(De)coding using simple combinatorics *vs.* sophisticated coding techniques (rank metric codes);
- Performs better at high cache memory regime *vs.* low cache memory regime.

*Largely unrelated ...*
*Are there any connections?*

# A Hidden Connection

Putting them together will give some insights,

- Yu scheme transmits (4 symbols):

$$\begin{array}{ll} A_{23} + A_{13} + B_{12}, & A_{24} + A_{14} + C_{12} \\ A_{34} + B_{14} + C_{13}, & A_{34} + B_{24} + C_{23} \end{array}$$

$$\Downarrow$$

- Tian-Chen scheme (9 symbols):

$$\begin{array}{lllllll} A_{23} + A_{13}, & B_{12}, & & A_{24} + A_{14}, & C_{12} \\ A_{34}, & B_{14}, & C_{13}, & & B_{24}, & C_{23} \end{array}$$

Question: full decomposition $\rightarrow$ partial decomposition?

# An Alternative Scheme

An alternative partial decomposition scheme:

- Choose some transmissions to decompose;
- From those transmissions, choose some files to decompose;
- E.g., $(N, K) = (3, 4), t = 2$,
  - Demand (A,A,B,C), transmits

$$\begin{array}{lll} A_{23} + A_{13} + B_{12}, & & A_{24} + A_{14} + C_{12} \\ A_{34}, \quad B_{14} + C_{13}, & & B_{24} + C_{23} \end{array}$$

# An Alternative Scheme

Apply this decomposition pattern to all degenerate demands:

- Demand (A,A,B,B), transmits

$$
\begin{array}{ll}
A_{23} + A_{13} + B_{12}, & A_{24} + A_{14} + B_{12} \\
A_{34}, \quad B_{14} + B_{13}, & \qquad B_{24} + B_{23}
\end{array}
$$

- Demand (A,A,A,C), transmits

$$
\begin{array}{ll}
A_{23} + A_{13} + A_{12}, & A_{24} + A_{14} + C_{12} \\
A_{34}, \quad A_{14} + C_{13}, & \qquad A_{24} + C_{23}
\end{array}
$$

- Demand (A,A,A,A), transmits

$$
\begin{array}{ll}
A_{23} + A_{13} + A_{12}, & A_{24} + A_{14} + A_{12} \\
A_{34}, \quad A_{14} + A_{13}, & \qquad A_{24} + A_{23}
\end{array}
$$

# An Alternative Scheme

- A good choice of decomposition: produces $(M, R) = (\frac{4}{3}, \frac{5}{6})$, a new corner poin;
- Decompose which transmissions and what files are "good"?
- First, we need a generalized framework to describe "decomposition".

# Partial Decomposition and A New Code Example

A formal description of partial decomposition:

- Demand vector $\boldsymbol{d} = \{d_1, \ldots, d_K\}$: demands by $K$ users, $d_k \in [1:N]$.
- Transmission type $\boldsymbol{t} = (t_1, \ldots, t_N)$: associated with a transmission in Yu scheme:

$$\oplus_{k \in \mathcal{B}} W_{d_k, \mathcal{B} \backslash k}, \quad |\mathcal{B}| = t + 1,$$

  $t_n$: the number of users in $\mathcal{B}$ requesting file $W_n$.

# Partial Decomposition and A New Code Example

- Decomposition pattern $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$: a partition of $\mathrm{supp}(\boldsymbol{t})$.
- The collection of all valid $\boldsymbol{t}$ is $\mathcal{T}_{\boldsymbol{d}}^{(t)}$.
- The collection of all $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}}$ is $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \{\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} | \boldsymbol{t} \in \mathcal{T}_{\boldsymbol{d}}^{(t)}\}$.
- The collection of all possible $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$ is $\mathfrak{P}_{\boldsymbol{t},\boldsymbol{d}}$.

# Partial Decomposition and A New Code Example

- E.g., $(N, K) = (3, 4), t = 2, \boldsymbol{d} = (A, A, B, C) = (1, 1, 2, 3)$:

  We have $\mathcal{T}_{\boldsymbol{d}}^{(t)} = \{(2, 1, 0), (2, 0, 1), (1, 1, 1)\}$, for example, consider $\boldsymbol{t} = (1, 1, 1)$:

  $$\begin{array}{lcl} \mathcal{B} = \{1, 3, 4\} : A_{34} + B_{14} + C_{13} & \xrightarrow{\text{decompose}} & A_{34},\ B_{14} + C_{13} \\ \mathcal{B} = \{2, 3, 4\} : A_{34} + B_{24} + C_{23} & & A_{34},\ B_{24} + C_{23} \end{array}$$

  then $\mathcal{P}_{\boldsymbol{t},\boldsymbol{d}} = \mathcal{P}_{(1,1,1)(1,1,2,3)} = \{\{1\}, \{2, 3\}\}$.

# Partial Decomposition and A New Code Example

- Code instance: instance is a "copy" of $N\binom{K}{t}$ file segments.
- Auxiliary variable $\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}$: a portion of all instances adopting decomposition pattern $\mathcal{P}_{\boldsymbol{d}}^{(t)}$,

$$\sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{\boldsymbol{t},\boldsymbol{d}}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1, \quad \boldsymbol{d} \in \mathcal{D},$$

$$1 \geq \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \geq 0, \quad \boldsymbol{d} \in \mathcal{D}, \quad \mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{\boldsymbol{t},\boldsymbol{d}}.$$

# Partial Decomposition and A New Code Example

The new partial decomposition coding scheme:

- Prefetching
  - Partition each file into $r\binom{K}{t}$ segments ($r$ instances each of $\binom{K}{t}$ segments), each in finite field $\mathbb{F}_{2^m}$;
  - Each user is allocated $P = rN\binom{K-1}{t-1}$ symbols;

# Partial Decomposition and A New Code Example

The new partial decomposition coding scheme:

- Prefetching
  - Encode $P$ symbols using $(P_o, P)$ systematic rank metric code, s.t.
    $$P_o - P = rM'_r\binom{K}{t} \quad (M'_r \text{ to be defined later}),$$
    caches the parity symbols;
  - Rank metric code existence condition: $m \geq P_o$.

# Partial Decomposition and A New Code Example

- Delivery:
  - Find all possible decomposition patterns $\mathfrak{P}_{\boldsymbol{t},\boldsymbol{d}}$;
  - Find how many coding instances use each pattern, i.e. $\alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}}$;
  - E.g., $(N, K) = (3, 4), t = 2$, the following code produces a good corner point.

# Partial Decomposition and A New Code Example

1) Demand $\boldsymbol{d}$ = (A,A,B,C)=(1,1,2,3), one decomposition pattern: $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)} = \boldsymbol{\mathcal{P}}_{(1,1,2,3)}^{(2)}$:

$$\begin{aligned}
&\mathcal{P}_{(2,1,0),(1,1,2,3)} = \{\{1,2\}\} = \{\{A,B\}\} : A_{23} + A_{13} + B_{12},\\
&\mathcal{P}_{(2,0,1),(1,1,2,3)} = \{\{1,3\}\} = \{\{A,C\}\} : A_{24} + A_{14} + C_{12},\\
&\mathcal{P}_{(1,1,1),(1,1,2,3)} = \{\{1\},\{2,3\}\} = \{\{A\},\{B,C\}\} :\\
&\qquad\qquad A_{34}, \quad B_{14} + C_{13},\; B_{24} + C_{23}
\end{aligned}$$

For this pattern,

$$R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 5, M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 8, k \in [1:4], \text{ we let } \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1.$$

# Partial Decomposition and A New Code Example

2) Demand $\boldsymbol{d}$ = (A,A,B,B)=(1,1,2,2), two decomposition patterns:
1[st] $\mathcal{P}_{\boldsymbol{d}}^{(t)}$: without any decomposition

$$\begin{array}{ll} A_{23}^{(1)} + A_{13}^{(1)} + B_{12}^{(1)}, & A_{24}^{(1)} + A_{14}^{(1)} + B_{12}^{(1)} \\ A_{34}^{(1)} + B_{14}^{(1)} + B_{13}^{(1)}, & A_{34}^{(1)} + B_{24}^{(1)} + B_{23}^{(1)} \end{array}$$

For this pattern,

$$R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 4, M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 9, k \in [1:4], \text{ we let } \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1/2.$$

# Partial Decomposition and A New Code Example

2) Demand $\boldsymbol{d}$ = (A,A,B,B)=(1,1,2,2), two decomposition patterns:
$2^{\text{nd}}$ $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$: as following

$$\begin{aligned}\mathcal{P}_{(2,1,0),(1,1,2,2)} = \{\{1\},\{2\}\} = \{\{A\},\{B\}\} : A_{23}^{(2)} + A_{13}^{(2)}, B_{12}^{(2)},\\ A_{24}^{(2)} + A_{14}^{(2)}, B_{12}^{(2)};\\ \mathcal{P}_{(1,2,0),(1,1,2,2)} = \{\{1\},\{2\}\} = \{\{A\},\{B\}\} : A_{34}^{(2)}, B_{14}^{(2)} + B_{13}^{(2)},\\ A_{34}^{(2)}, B_{24}^{(2)} + B_{23}^{(2)};\end{aligned}$$

For this pattern,

$$R_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 6, M_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)},k} = 7, k \in [1:4], \text{ we let } \alpha_{\boldsymbol{d},\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}} = 1/2.$$

# Partial Decomposition and A New Code Example

3) Demand $\boldsymbol{d}$ = (A,A,A,A)=(1,1,1,1), two decomposition patterns:
$1^{\text{st}}$ $\boldsymbol{\mathcal{P}}_{\boldsymbol{d}}^{(t)}$: without any decomposition

$$A_{23}^{(i)} + A_{13}^{(i)} + A_{12}^{(i)}, \quad A_{24}^{(i)} + A_{14}^{(i)} + A_{12}^{(i)}$$
$$A_{34}^{(i)} + A_{14}^{(i)} + A_{13}^{(i)}, \quad \text{for } i = 1, 2, 3, 4, 5.$$

For this pattern,

$$R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 3, M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 9, k \in [1:4], \text{ we let } \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 5/6.$$

# Partial Decomposition and A New Code Example

3) For demand $\boldsymbol{d}$ = (A,A,A,A)=(1,1,1,1), two decomposition patterns:
2nd $\mathcal{P}_{\boldsymbol{d}}^{(t)}$: special uncoded decomposition pattern

$$A_{23}^{(6)}, A_{13}^{(6)}, A_{24}^{(6)}, A_{14}^{(6)}, A_{12}^{(6)}, A_{34}^{(6)}$$
$$B_{23}^{(6)}, B_{13}^{(6)}, B_{24}^{(6)}, B_{14}^{(6)}, B_{12}^{(6)}, B_{34}^{(6)}$$

For this pattern,

$$R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 12, M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} = 3, k \in [1:4], \text{ we let } \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1/6.$$

# Partial Decomposition and A New Code Example

- The above example uses $r = 6$ instances of code,

| Demand | $(1,1,2,3)$ | $(1,1,2,2)$ | | $(1,1,1,2)$ | | $(1,1,1,1)$ | |
|---|---|---|---|---|---|---|---|
| Pattern | 1 | 1 | 2 | 1 | 2 | 1 | 2 |
| $(M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k}, R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}})$ | $(8,5)$ | $(9,4)$ | $(7,6)$ | (9,4) | (7,6) | $(9,3)$ | $(3,12)$ |
| $r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}$ | 6 | 3 | 3 | 3 | 3 | 5 | 1 |
| $\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}$ | 1 | $1/2$ | $1/2$ | $1/2$ | $1/2$ | $5/6$ | $1/6$ |

Memory-rate pair

$$(M, R) = \frac{1}{r\binom{K}{t}}\left(\max_{k\in[1:K]} \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k}, \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\right)$$

would be $(\frac{4}{3}, \frac{5}{6})$.

# A New Information-Theoretic Inner Bound

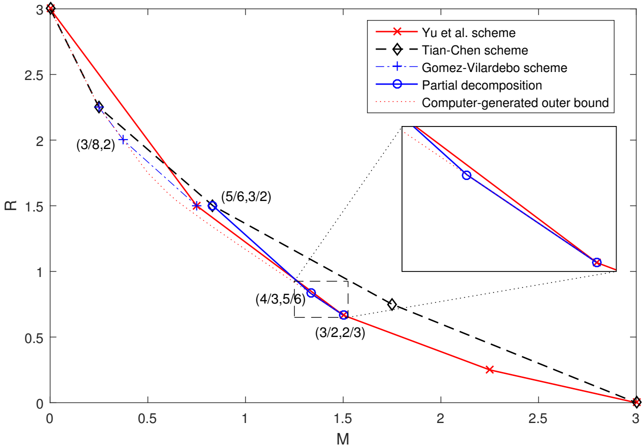



Figure: A new information-theoretic inner-bound for the example caching system $(N, K) = (3, 4)$, $t = 2$.

# A New Information-Theoretic Inner Bound

- Special uncoded transmission to keep the decomposition rule;
- As system parameters grow, no. of decomposition patterns will be huge;
- Hand-calculate best decompositions for all $\boldsymbol{d}, \boldsymbol{t}$? Impossible.

We need a *computer-aided* approach.

# A New Information-Theoretic Inner Bound

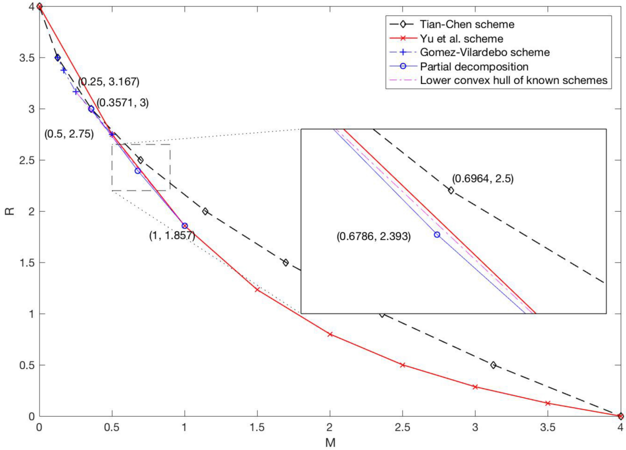



Figure: A new information-theoretic inner-bound for the example caching system $(N, K) = (4, 8), t = 2$.

# A Linear Programming Framework

Searching optimal decomposition patterns by linear programming:

- Rate region $\mathcal{R}^{(t)}$: collection of memory-rate pairs $(M, R)$ s.t. there exists set of $\{\alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\}$ satisfying

$$\sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{t,\boldsymbol{d}}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} = 1, \quad \boldsymbol{d} \in \mathcal{D},$$

$$1 \geq \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \geq 0, \quad \boldsymbol{d} \in \mathcal{D}, \quad \mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{t,\boldsymbol{d}},$$

$$\sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{t,\boldsymbol{d}}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \leq R\binom{K}{t}, \quad \boldsymbol{d} \in \mathcal{D},$$

$$\sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)} \in \mathfrak{P}_{t,\boldsymbol{d}}} \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k} \leq M\binom{K}{t}, \quad \boldsymbol{d} \in \mathcal{D}, k \in [1:K].$$

# A Linear Programming Framework

- Further define

$$(M'_{\boldsymbol{d}}, R'_{\boldsymbol{d}}) = \frac{1}{r\binom{K}{t}} \left( \max_{k\in[1:K]} \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} M_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)},k}, \sum_{\mathcal{P}_{\boldsymbol{d}}^{(t)}} r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} R_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \right)$$

and

$$M'_r \triangleq \max_{\boldsymbol{d}\in\mathcal{D}} M'_{\boldsymbol{d}}, \qquad R'_r \triangleq \max_{\boldsymbol{d}\in\mathcal{D}} R'_{\boldsymbol{d}}.$$

# A Linear Programming Framework

- Choose number of instances $r$ s.t. exist integers $\{r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}\}$

$$\left| \frac{r_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}}}{r} - \alpha_{\boldsymbol{d},\mathcal{P}_{\boldsymbol{d}}^{(t)}} \right| \leq \epsilon,$$

  s.t. $\epsilon \geq 0$ and arbitrarily small, we have

$$\lim_{r \to \infty} (M'_r, R'_r) = (M, R)$$

  be the effective memory-rate pair of the new code.

# Conclusion

- A single scheme unifying two general classes of schemes;
- Yu scheme and Tian-Chen scheme are two extreme points;
- Transmission type plays an important role and was overlooked before;
- Performance improvement not large, reside in the middle $M$ regime $\rightarrow$ not a surprise.

# Conclusion

- The new coding scheme cannot incorporate Gómez-Vilardebó[1]'s scheme, a more generalized code may exist?
- Simplify linear programming: identify and remove those "bad" decomposing patterns?

[1] J. Gómez-Vilardebó, "Fundamental limits of caching: Improved bounds with coded prefetching," *arXiv:1612.09071*, Dec. 2016.