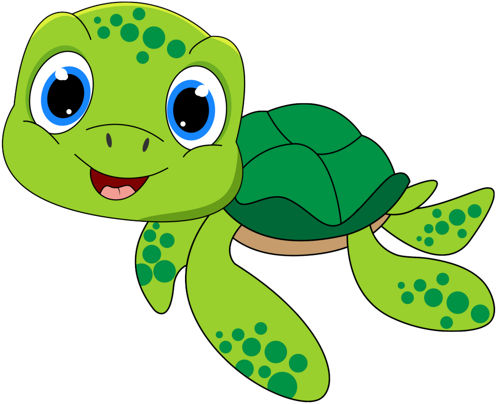
Codes for (Un)Expected Loads

Emina Soljanin, Rutgers



Acknowledgments

Collaborators:

G. Alfarano, M. Aktaș, S. Andersen, A. Johnston, G. Joshi, S. Kadhe, E. Karimi, F. Kazemi, A. Kılıç, O. Kolosev, **H. Ly Minh**, G. Matthews, C. Mayer, A. Ravagnani, A. Sprintson, G. Yadgar

Funding:

NSF Award # 2122400: Service Rates of Codes

ICERM WiSDM programs in 2017 and 2019

Distributed Service Model

There are n nodes providing service to multiple concurrent users, e.g., cloud edge nodes providing streaming, download, computing.

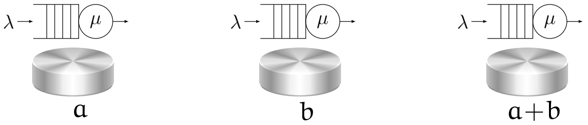
We distinguish between two functional components at each node: one for data storage and the other for service request processing.

Distributed Service Model

There are n nodes providing service to multiple concurrent users, e.g., cloud edge nodes providing streaming, download, computing.

We distinguish between two functional components at each node: one for data storage and the other for service request processing.

Example:
Three nodes provide data-download service to multiple concurrent users, where each user wants either data object a or data object b.



Data Storage Model

Simple Redundant Storage

- k equal size data objects are stored across n servers ($k \leqslant n$).
- Data objects are represented as elements of $\mathbb{F}_q$.
- Coded object are linear combinations of data specified by $\mathbb{F}_q^k$ vectors.
- Each server stores a single coded object (one of n).

$\Longrightarrow$ A data object can be recovered from multiple sets of coded objects.

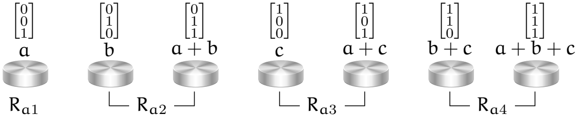
Data Storage Model

Simple Redundant Storage

- k equal size data objects are stored across n servers ($k \leqslant n$).
- Data objects are represented as elements of $\mathbb{F}_q$.
- Coded object are linear combinations of data specified by $\mathbb{F}_q^k$ vectors.
- Each server stores a single coded object (one of n).

$\Longrightarrow$ A data object can be recovered from multiple sets of coded objects.

Example: Data objects a, b, and c stored across $n = 7$ nodes:

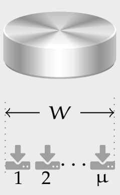


$\Longrightarrow$ a can be recovered from any of the sets $R_{a1}, R_{a2}, R_{a3}, R_{a4}$.

Data Service and Request Models

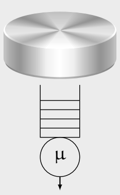
Different practical service models are mathematically equivalent.

For service, we consider the bandwidth and the queuing model:



stores one data object

Server's bandwidth W can accomodate up to μ users.



stores one data object

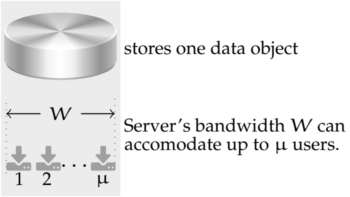
Users queue for download.

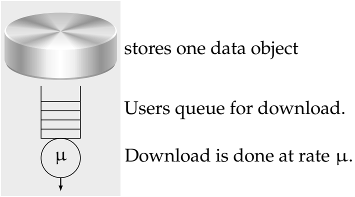
Download is done at rate μ.

Data Service and Request Models

Different practical service models are mathematically equivalent.

For service, we consider the bandwidth and the queuing model:





Requests for objects i, $i \in \{1, \ldots, k\}$:

- In the queuing model, requests for object i arrive at rate λ_i.
- In the bandwidth model, the number of requests for object i is λ_i

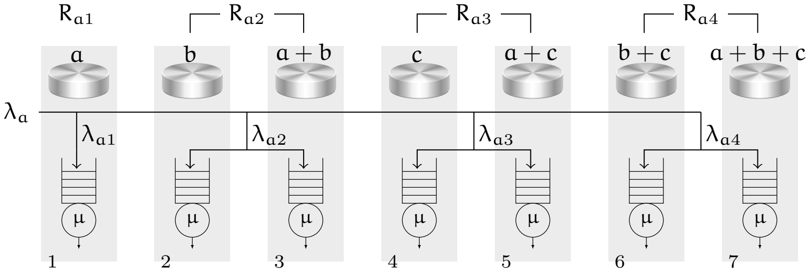
$\implies$

A server can handle multiple download requests.

Distributed Service Model – An Example

λ_a is the request rate (demand) for object a

λ_{aj} is the portion of λ_a assigned to the recovery set R_{aj}, $j \in \{1, 2, 3, 4\}$.

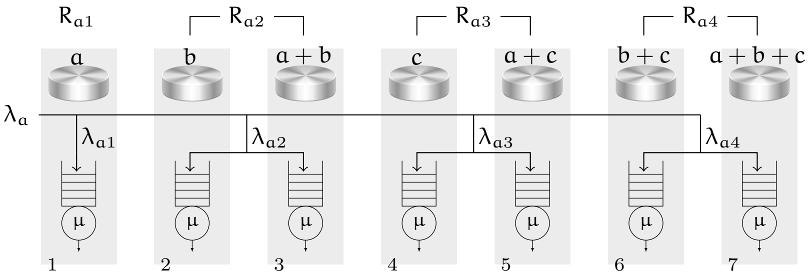


$\{\lambda_{a1}, \lambda_{a2}, \lambda_{a3}, \lambda_{a4}\}$ is a request allocation for λ_a.

Distributed Service Model – An Example

λ_a is the request rate (demand) for object a

λ_{aj} is the portion of λ_a assigned to the recovery set R_{aj}, $j \in \{1, 2, 3, 4\}$.



$\{\lambda_{a1}, \lambda_{a2}, \lambda_{a3}, \lambda_{a4}\}$ is a request allocation for λ_a.

Which request vectors $(\lambda_a, \lambda_b, \lambda_c)$ can be serviced by the system?

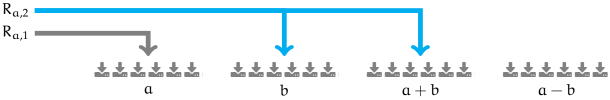
Codes with Locality and Availability in Service

What are the simultaneous recovery sets for a in the following code?

Codes with Locality and Availability in Service

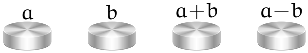
What are the simultaneous recovery sets for a in the following code?

- There are two a-recovery sets in the LRC/availability model, e.g.,

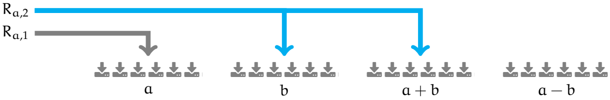


Codes with Locality and Availability in Service

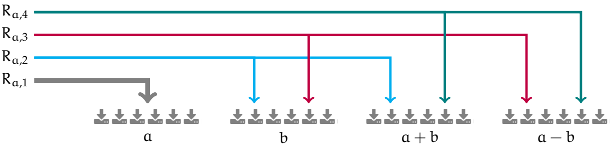
What are the simultaneous recovery sets for a in the following code?



- There are two a-recovery sets in the LRC/availability model, e.g.,



- There are four a-recovery sets in the service model:



Service Rate Region – A Polytope in $\mathbb{R}^k$

Set of vectors $(\lambda_1, \ldots, \lambda_k)$ that can be served by the system

λ_i is the request rate (demand) for object i, $i = 1, \ldots, k$.

λ_{ij} is the portion of λ_i assigned to the recovery set R_{ij}, $j = 1, \ldots, t_i$.

The request vector $(\lambda_1, \ldots, \lambda_k)$ can be serviced by the system iff there exist λ_{ij} satisfying the following constraints:

1. No server is assigned requests in excess of its service rate:

$$\sum_{i=1}^{k} \sum_{\substack{1 \leqslant j \leqslant t_i \\ \ell \in R_{ij}}} \lambda_{ij} \leqslant \mu \quad \text{for} \quad 1 \leqslant \ell \leqslant n.$$

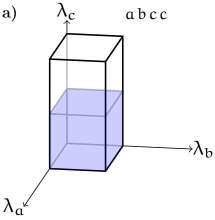
2. All objects' requests are served: $\sum_{j=1}^{t_i} \lambda_{ij} = \lambda_i$ for $1 \leqslant i \leqslant k$

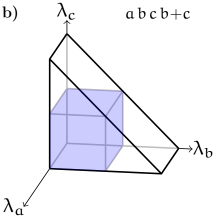
$\{\lambda_{ij} : 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant t_i\}$ is a request allocation for $(\lambda_1, \ldots, \lambda_k)$.

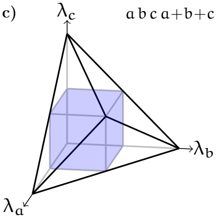
If we require that λ_{ij} be either 0 or μ, we speak of integral service rates.

Three Storage Schemes and Their Service Rates

$k = 3$ data objects stored across $n = 4$ nodes







Many (kinds of) questions are of interest.

Service Rate Region Problem(s) Formulation

System Model:

- k data objects are stored redundantly across n nodes.
- Data objects are represented as elements of some finite field.
- Each server stores a <u>linear combination</u> of data objects, i.e., a coded object of the same size (same field).
- Requests for object i, $i \in \{1, \ldots, k\}$ arrive to the system at rate λ_i.
- At each node, requests are serviced at rate $\mu = 1$.

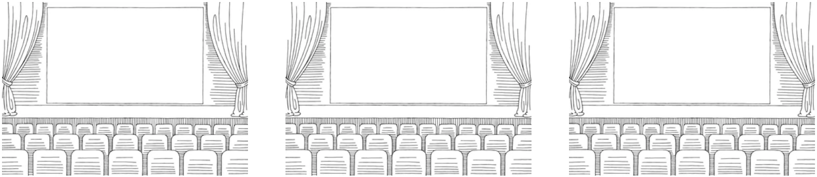
SOME OBJECTIVES:

1. <u>Determine the set of rates</u> $(\lambda_1, \ldots, \lambda_k)$  that can be supported by the system implementing some common redundancy scheme.
2. <u>Design a redundancy scheme</u> in order to maximize and/or shape the of region of supported arrival rates under some limited resources.
3. <u>Evaluate the system's performance</u> for a given stochastic model of $(\lambda_1, \ldots, \lambda_k)$ (e.g., probability of supported rates, load imbalance).

A Problem of City with Two Movies & Three Cinemas

- There are two movies a & b and three cinemas with 100 seats each.
- λ_a people want to see a and λ_b people want to see b.
- We know that the city's population is 200 $\implies \lambda_a + \lambda_b \leqslant 200$

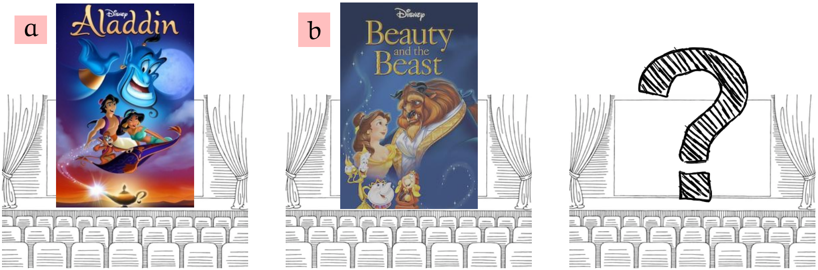
Q1: Which movie should each cinema play?



A Problem of City with Two Movies & Three Cinemas

- There are two movies a & b and three cinemas with 100 seats each.
- λ_a people want to see a and λ_b people want to see b.
- We know that the city's population is 200 $\implies \lambda_a + \lambda_b \leqslant 200$

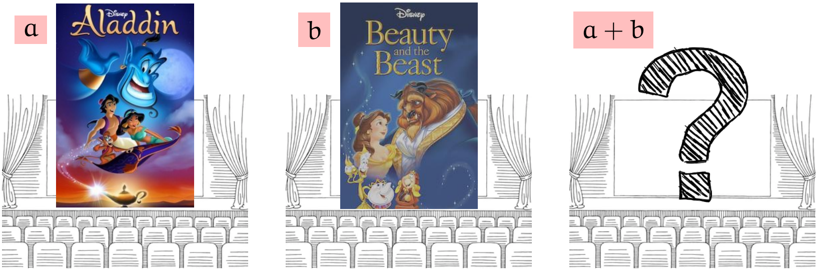
Q1: Which movie should each cinema play?



A Problem of City with Two Movies & Three Cinemas

- There are two movies a & b and three cinemas with 100 seats each.
- λ_a people want to see a and λ_b people want to see b.
- We know that the city's population is 200 $\implies \lambda_a + \lambda_b \leqslant 200$

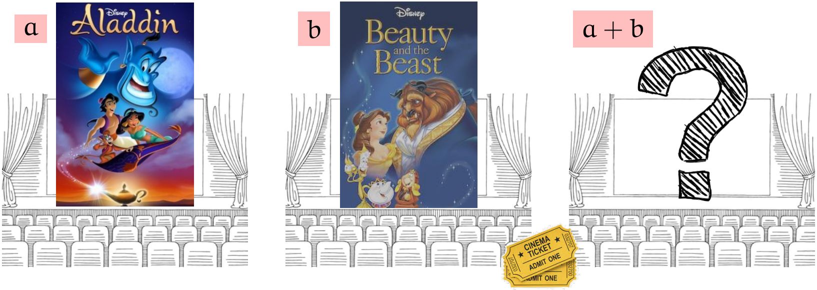
Q1: Which movie should each cinema play?



A Problem of City with Two Movies & Three Cinemas

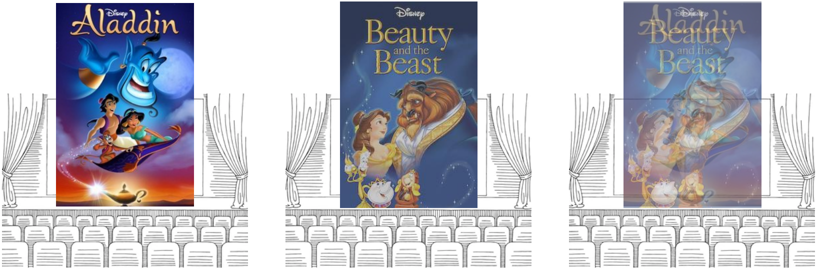
- There are two movies a & b and three cinemas with 100 seats each.
- λ_a people want to see a and λ_b people want to see b.
- We know that the city's population is 200 $\implies \lambda_a + \lambda_b \leqslant 200$

Q1: Which movie should each cinema play?



A Problem of City with Two Movies & Three Cinemas

a | b | $a+b$



Q2: Can λ_a people see a and λ_b people see b as long as $\lambda_a + \lambda_b \leqslant 200$?

Q3: Which vectors (λ_a, λ_b) are possible for a given redundancy scheme?

“Matrix” G - A Collection of Storage Specifying Columns

G is a $k \times n$, $k \leqslant n$, rank-k matrix &

 Columns of G are a multi-set V of points in $\mathbb{PG}(k-1, q)$.

“Matrix” G - A Collection of Storage Specifying Columns

G is a $k \times n$, $k \leqslant n$, rank-k matrix &


Columns of G are a multi-set V of points in $\mathbb{PG}(k-1, q)$.

Example #1 – the [7, 3] binary Simplex code

Matrix $G = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ encodes data $[c \;\; b \;\; a]$ as follows:

$$[c \;\; b \;\; a] \cdot G = [a \;\; b \;\; a{+}b \;\; c \;\; a{+}c \;\; b{+}c \;\; a{+}b{+}c]$$

“Matrix” G - A Collection of Storage Specifying Columns

G is a $k \times n$, $k \leqslant n$, rank-k matrix &

Columns of G are a multi-set V of points in $\mathbb{PG}(k-1, q)$.

Example #1 – the $[7,3]$ binary Simplex code

Matrix $G = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ encodes data $[c \;\; b \;\; a]$ as follows:

$$[c \;\; b \;\; a] \cdot G = [a \;\; b \;\; a{+}b \;\; c \;\; a{+}c \;\; b{+}c \;\; a{+}b{+}c]$$

Example #2 – two $[4,2]$ MDS codes (α is a primitive in $\mathbb{F}_q$, $q > 4$.)

$$G_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix}$$

These storage schemes provide different data access performance.

Minimal Recovery Sets

Subset R of columns in G is a minimal recovery set of data object a if

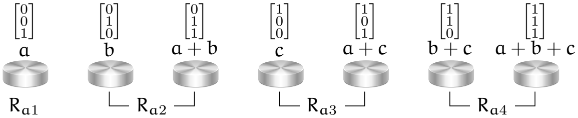
- $a \in \text{span}(R)$
- $S \subset R \implies a \notin \text{span}(S)$

Minimal Recovery Sets

Subset R of columns in G is a minimal recovery set of data object a if

- $a \in \text{span}(R)$
- $S \subset R \implies a \notin \text{span}(S)$

Example:



R_{a1}, R_{a2}, R_{a3}, and R_{a4} are the recovery sets of size one and two of a in

$$G = \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$


There is a minimal recovery set for a of size three!

Minimal Recovery Sets

Subset R of columns in G is a <u>recovery set of e_j</u> if

- $e_j \in \mathrm{span}(R)$
- $S \subset R \implies e_j \notin \mathrm{span}(S)$

<u>Example:</u>

Let $G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix}$. Then

- the (size one and two) recovery sets of $e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ are

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ \alpha \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ \alpha \end{bmatrix} \right\}$$

- the (size one and two) recovery sets of $e_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are

$$\left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ \alpha \end{bmatrix} \right\} \; \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ \alpha \end{bmatrix} \right\}$$

 The recovery sets do not have to be disjoint (as for LRCs).

The Recovery Graph Γ_G of a $2 \times n$ Matrix G

and other matrices with only size-2 recovery sets

NODES:

Γ_G has n nodes corresponding to the columns of G, and
i additional nodes $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]_j$ for systematic columns $j = 1, \ldots, i$.

EDGES:

If two nodes correspond to a recovery set of e_j,
they are connected by an edge which is given label e_j.

The Recovery Graph Γ_G of a $2 \times n$ Matrix G

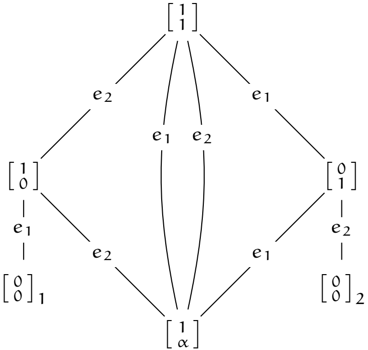
and other matrices with only size-2 recovery sets

NODES:
Γ_G has n nodes corresponding to the columns of G, and
i additional nodes $\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]_j$ for systematic columns $j = 1, \ldots, i$.

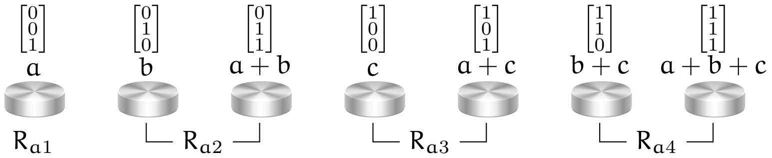
EDGES:
If two nodes correspond to a recovery set of e_j,
they are connected by an edge which is given label e_j.

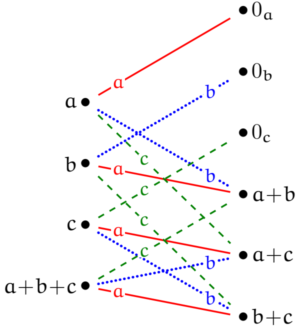
Example:

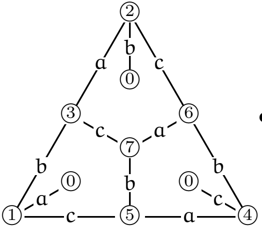
$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix}$$



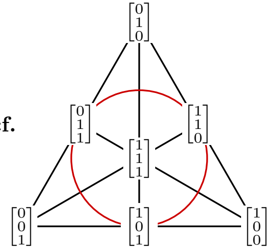
Recovery Graphs of Binary Simplex Codes







cf.



Fractional Matching & Service Rates on Recovery Graphs

A fractional matching in (V, E) is a vector $\boldsymbol{w} \in \mathbb{R}^{|E|}$ whose components (weights) w_ϵ, $\epsilon \in E$, are non-negative and $\sum_{\epsilon \ni v} w_\epsilon \leqslant 1$ for each $v \in V$.

Fractional Matching & Service Rates on Recovery Graphs

A fractional matching in (V, E) is a vector $\boldsymbol{w} \in \mathbb{R}^{|E|}$ whose components (weights) w_ϵ, $\epsilon \in E$, are non-negative and $\sum_{\epsilon \ni v} w_\epsilon \leqslant 1$ for each $v \in V$.

We define $\lambda_j^{\boldsymbol{w}}$, the service rate for e_j in matching $\boldsymbol{w}$ as the sum of weights w_ϵ of all e_j-labeled edges $\epsilon \in E$. We say that $\boldsymbol{w}$ yields $\lambda_j^{\boldsymbol{w}}$.

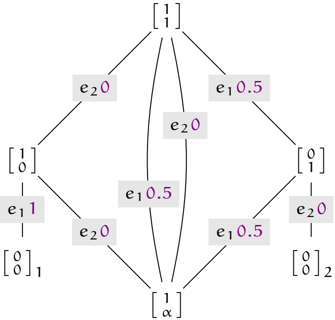
Fractional Matching & Service Rates on Recovery Graphs

A fractional matching in (V, E) is a vector $\boldsymbol{w} \in \mathbb{R}^{|E|}$ whose components (weights) w_ϵ, $\epsilon \in E$, are non-negative and $\sum_{\epsilon \ni v} w_\epsilon \leqslant 1$ for each $v \in V$.

We define $\lambda_j^{\boldsymbol{w}}$, the service rate for e_j in matching $\boldsymbol{w}$ as the sum of weights w_ϵ of all e_j-labeled edges $\epsilon \in E$. We say that $\boldsymbol{w}$ yields $\lambda_j^{\boldsymbol{w}}$.

A matching with service rates $\lambda_1 = 2.5$ and $\lambda_2 = 0$.

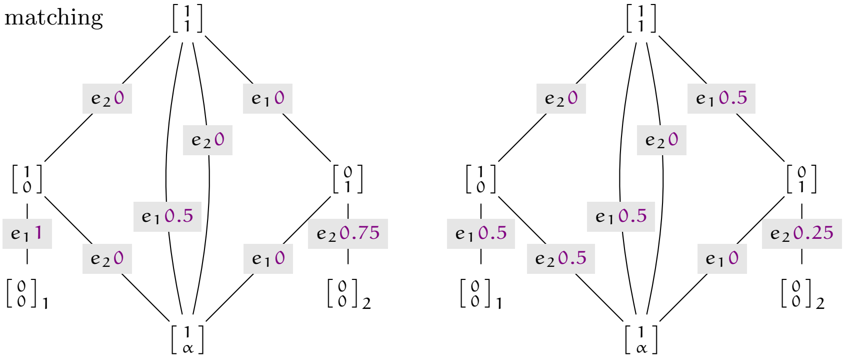
$$G_2(4,2) = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix} \longrightarrow \Gamma_2(4,2)$$



A Service Vector and its Matchings

Two matchings for $(1.5, 0.75)$ service vector in $\Gamma_2(4,2)$:

greedy matching



Fractional Matching and Service Polytopes of $\Gamma_G = (V, E)$

<u>The set of all fractional matchings</u> in $\Gamma_G = (V, E)$ is a polytope in $\mathbb{R}^{|E|}$, called the **fractional matching polytope** and denoted by $\mathrm{FMP}(\Gamma_G)$.

<u>The set of all service vectors</u> in $\Gamma_G = (V, E)$ is a polytope in $\mathbb{R}^k$. We call it the **service rate region**, and denote by $\mathcal{R}(\Gamma_G)$.

$\implies$ $\mathcal{R}(\Gamma_G)$ is the image of $\mathrm{FMP}(\Gamma_G)$ under a linear map from $\mathbb{R}^{|E|}$ to $\mathbb{R}^k$.

Fractional Matching and Service Polytopes of $\Gamma_G = (V, E)$

The set of all fractional matchings in $\Gamma_G = (V, E)$ is a polytope in $\mathbb{R}^{|E|}$, called the **fractional matching polytope** and denoted by $\mathrm{FMP}(\Gamma_G)$.

The set of all service vectors in $\Gamma_G = (V, E)$ is a polytope in $\mathbb{R}^k$. We call it the **service rate region**, and denote by $\mathcal{R}(\Gamma_G)$.

$\implies$ $\mathcal{R}(\Gamma_G)$ is the image of $\mathrm{FMP}(\Gamma_G)$ under a linear map from $\mathbb{R}^{|E|}$ to $\mathbb{R}^k$.

The service rate region problem:

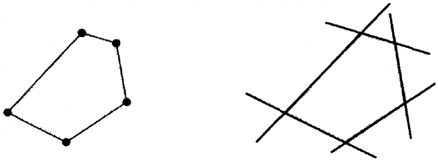
$$\boxed{\text{Find } \mathcal{R}(\Gamma_G) = \Big\{\boldsymbol{\lambda}^{\boldsymbol{w}} \in \mathbb{R}^k : \boldsymbol{w} \in \mathrm{FMP}(\Gamma_G), \lambda_i^{\boldsymbol{w}} = \sum_{\epsilon \text{ labeled by } e_i} w_\epsilon \Big\}.}$$

What does it mean to characterize a polytope?

We need to specify points or hyperplanes!

Convex Polytope in $\mathbb{R}^{\mathrm{d}}$:

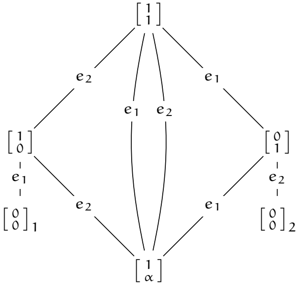
- convex hull of a finite set of points OR
- intersection of finitely many closed half spaces that is bounded

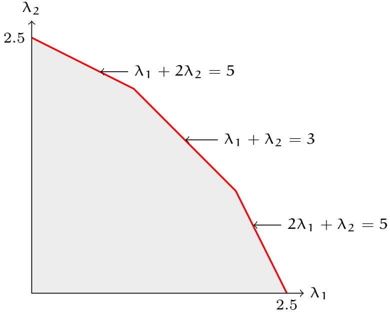


Ziegler's Lectures on Polytopes, Springer, GTM 152, revised first edition.

A Service Rate Polytope Example

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix} \longrightarrow \Gamma_G$$

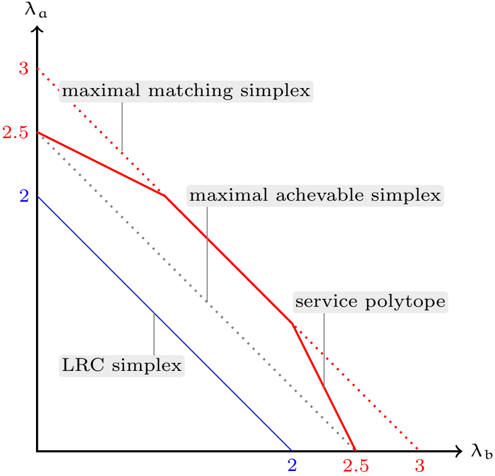




Some bounds on $\mathcal{R}(\Gamma_G)$ follow from the fractional matching number of Γ_G.

Two Bounding Simplexes (and the LRC Simplex)

Bounds on $\sum_{i=1}^{k} \lambda_i$ for any vertex $(\lambda_1 \ldots, \lambda_k)$ of the service region



We often compare service rate regions based on their bounding simplexes.

The Maximal Matching Simplex

Fractional Matching and Vertex Cover Numbers

For matching $\boldsymbol{w}$, we have $\displaystyle\sum_{i=1}^{k} \lambda_i^{\boldsymbol{w}} = \sum_{\epsilon \in E} w_\epsilon$, $\longleftarrow$ the size of $\boldsymbol{w}$.

$\Longrightarrow$

For all $\boldsymbol{\lambda} \in \mathcal{R}(\Gamma_G)$, we have $\displaystyle\sum_{i=1}^{k} \lambda_i \leqslant \max_{\boldsymbol{w} \in FMP(\Gamma_G)} \sum_{\epsilon \in E} w_\epsilon = \nu^*(\Gamma_G)$

$\nu^*(\Gamma_G)$ is the fractional matching number of Γ_G.

$FMP(\Gamma_G) = \{\boldsymbol{w} \in \mathbb{R}^{|E|} : A\boldsymbol{w} \leqslant \mathbf{1},\ \boldsymbol{w} \geqslant 0\} \Longrightarrow$ finding $\nu^*(\Gamma_G)$ is an LP problem.

The Maximal Matching Simplex

Fractional Matching and Vertex Cover Numbers

For matching $\boldsymbol{w}$, we have $\displaystyle\sum_{i=1}^{k} \lambda_i^{\boldsymbol{w}} = \sum_{\epsilon \in E} w_\epsilon, \longleftarrow$ the size of $\boldsymbol{w}$.

$\Longrightarrow$

For all $\boldsymbol{\lambda} \in \mathcal{R}(\Gamma_G)$, we have $\displaystyle\sum_{i=1}^{k} \lambda_i \leqslant \max_{\boldsymbol{w} \in FMP(\Gamma_G)} \sum_{\epsilon \in E} w_\epsilon = \nu^*(\Gamma_G)$

$\nu^*(\Gamma_G)$ is the fractional matching number of Γ_G.

$FMP(\Gamma_G) = \left\{\boldsymbol{w} \in \mathbb{R}^{|E|} : A\boldsymbol{w} \leqslant \mathbf{1},\ \boldsymbol{w} \geqslant \mathbf{0}\right\} \Longrightarrow$ finding $\nu^*(\Gamma_G)$ is an LP problem.

Its dual finds the fractional vertex cover number $\tau^*(\Gamma_G) = \nu^*(\Gamma_G)$:

$$\tau^*(\Gamma_G) = \min \sum_{v \in V} \omega_v \text{ s.t. } A^T \boldsymbol{\omega} \geqslant \mathbf{1},\ \boldsymbol{\omega} \geqslant \mathbf{0}$$

A fractional vertex cover in (V, E) is a vector $\boldsymbol{\omega} \in \mathbb{R}^{|V|}$ whose components (weights) ω_v, $v \in V$, are non-negative and $\sum_{v \in \epsilon} \omega_v \geqslant 1$ for each $\epsilon \in E$.

The Maximal Matching Simplex

Axes Intercept Points of $\mathcal{R}(\Gamma_G)$

Two easy-to-prove observations:

1. Let $\lambda_j^{\max} = \max_{\boldsymbol{\lambda} \in \mathcal{R}(\Gamma_G)} \lambda_j$. Then $\lambda_j^{\max} e_j$ is a vertex of $\mathcal{R}(\Gamma_G)$.
2. Let Γ_G^j be the sub-graph of Γ_G induced by e_j-edges. Then $\lambda_j^{\max}$ is equals to the **matching number** of Γ_G^j.

The Maximal Matching Simplex

Axes Intercept Points of $\mathcal{R}(\Gamma_G)$

Two easy-to-prove observations:

1. Let $\lambda_j^{\max} = \max_{\boldsymbol{\lambda} \in \mathcal{R}(\Gamma_G)} \lambda_j$. Then $\lambda_j^{\max} e_j$ is a vertex of $\mathcal{R}(\Gamma_G)$.
2. Let Γ_G^j be the sub-graph of Γ_G induced by e_j-edges. Then $\lambda_j^{\max}$ is equals to the **matching number** of Γ_G^j.

The convex hull of points 0 and $\lambda_j^{\max} e_j$ is a k-simplex within $\mathcal{R}(\Gamma_G)$.

Bounding Simplexes for $G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & \alpha \end{bmatrix}$

<u>maximal matching simplex</u>:

intersection of half spaces

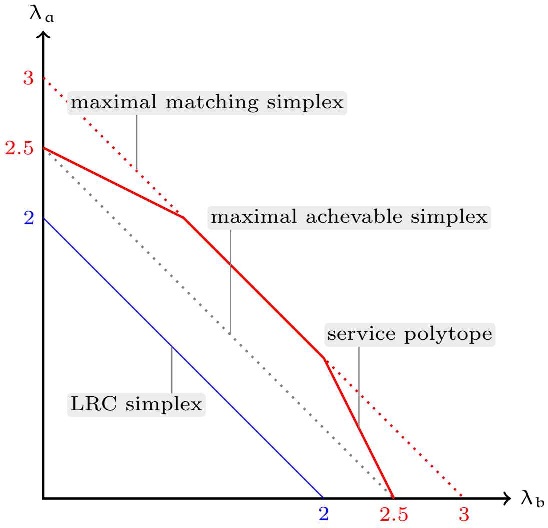
$\sum_{i=1}^{k} \lambda_i \leqslant \nu^*(\Gamma_G)$ & $\boldsymbol{\lambda} \geqslant 0$.

(Some points are achievable!)

<u>maximal achievable simplex:</u>

convex hull of points

$\lambda_i^{\max} e_i, \; i = 1, \ldots, k, \;$ & $\boldsymbol{\lambda} = 0$.



For binary simplex codes, these simplexes coincide.

Binary Simplex Codes and their Recovery Graphs

aka Hadamard Codes in CS literature

G_k consist of all distinct nonzero vectors of $\mathbb{F}_2^k$.
$\implies$ Γ_k nodes are labeled by k-bit stings with even and odd weight.

Binary Simplex Codes and their Recovery Graphs

aka Hadamard Codes in CS literature

G_k consist of all distinct nonzero vectors of $\mathbb{F}_2^k$.
$\implies$ Γ_k nodes are labeled by k-bit stings with even and odd weight.

Lemma: Structure of the recovery graph Γ_k:

1. Γ_k is bipartite. Edges connect odd-weight with odd-weight nodes.
2. Each odd-weight node of Γ_k has degree k.
3. The 2^{k-1} odd-weight nodes of form a minimum vertex cover of Γ_k.

Service Rate Region of $[2^k - 1, k]$ Binary Simplex Codes

Theorem: A simplex in $\mathbb{R}^k$!

The service region of the $[2^k - 1, k]$ binary Simplex code is defined by

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k \leqslant 2^{k-1}, \quad \lambda_i \geqslant 0, \ i = 1, \ldots, k.$$

Service Rate Region of $[2^k - 1, k]$ Binary Simplex Codes

Theorem: A simplex in $\mathbb{R}^k$!

The service region of the $[2^k - 1, k]$ binary Simplex code is defined by

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k \leqslant 2^{k-1}, \quad \lambda_i \geqslant 0, \ i = 1, \ldots, k.$$

Proof Sketch for the Achevability:

Fractional matching on the recovery graph that assigns weight $\lambda_i / 2^{k-1}$ to each e_i-labeled edge gives $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 2^{k-1}$.

Service Rate Region of $[2^k - 1, k]$ Binary Simplex Codes

Theorem: A simplex in $\mathbb{R}^k$!

The service region of the $[2^k - 1, k]$ binary Simplex code is defined by

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k \leqslant 2^{k-1}, \quad \lambda_i \geqslant 0, \; i = 1, \ldots, k.$$

Proof Sketch for the Achevability:

Fractional matching on the recovery graph that assigns weight $\lambda_i / 2^{k-1}$ to each e_i-labeled edge gives $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 2^{k-1}$.

Proof Sketch for the Converse:

For bipartite graphs, the size of the minimum vertex cover (here 2^{k-1}) is equal to the (fractional) matching number.

A Class of MDS Matrices $G_i(n,k)$, $i = 0,1,2,\dots,k$

α is a primitive in $\mathbb{F}_q$, $q > n$

$$G_0(n,k) = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \dots & \alpha^{i-1} & \dots & \alpha^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \dots & \alpha^{(i-1)(k-1)} & \dots & \alpha^{(n-1)(k-1)} \end{bmatrix},$$

$$G_i(n,k) = \left[\begin{array}{cccc|ccc} 1 & 0 & \dots & 0 & 1 & \dots & 1 \\ 0 & 1 & \dots & 0 & \alpha^i & \dots & \alpha^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & \alpha^{(i-1)i} & \dots & \alpha^{(i-1)(n-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \underbrace{0 \quad 0}_{} & & \dots & 0 & \alpha^{(k-1)i} & \dots & \alpha^{(k-1)(n-1)} \end{array}\right] \text{ for } i = 1,\dots,k.$$

$$\underbrace{\qquad\qquad\qquad}_{i \text{ columns of } I_k} \quad \underbrace{\qquad\qquad\qquad\qquad}_{n-i \text{ columns of } G_0(n,k)}$$

We denote the j-th column of I_k by e_j and call it systematic, $j = 1,\dots,k$.

Quasi-Uniform Recovery Hypergraphs

A hypergraph is a pair (V, E), where

- V is a finite set, the set of of vertices &
- E is a multiset of subsets of V called edges.

Uniform and Quasi-uniform Hypergraphs

We say that a hypergraph is

- k-uniform if each of its edges has size k
- (k, m)-quasi-uniform if each of its edges has either size k or m.

$\implies$ Graphs are 2-uniform hypergraphs.

Quasi-Uniform Recovery Hypergraphs

A hypergraph is a pair (V, E), where

- V is a finite set, the set of of vertices &
- E is a multiset of subsets of V called edges.

Uniform and Quasi-uniform Hypergraphs

We say that a hypergraph is

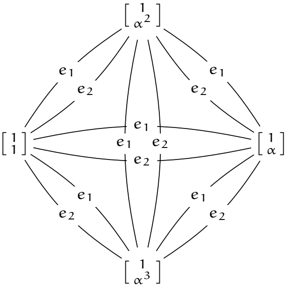
- k-uniform if each of its edges has size k
- (k, m)-quasi-uniform if each of its edges has either size k or m.

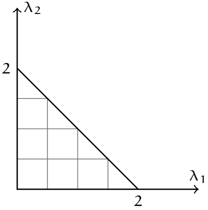
$\Longrightarrow$ Graphs are 2-uniform hypergraphs.

Q: What about recovery graphs $\Gamma_i(n, k)$ of $G_i(n, k)$, $i = 0, 1, 2, \ldots, k$?
A: They are $(k, 2)$-quasi-uniform and $\Gamma_0(n, k)$ is k-uniform.
The systematic edges have size 2; all other edges have size k.

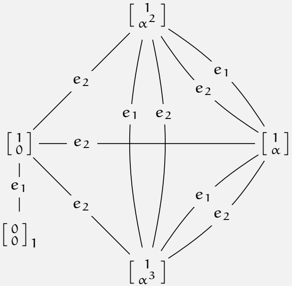
$[4, 2]$ MDS Matrices, Recovery Graphs, and Rate Regions

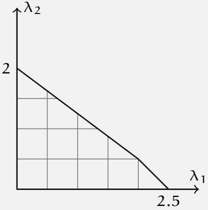
$$G_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 \end{bmatrix}$$



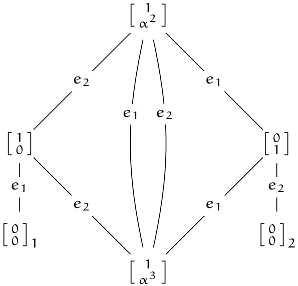


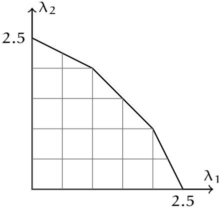
$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \alpha & \alpha^2 & \alpha^3 \end{bmatrix}$$





$$G_2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & \alpha^2 & \alpha^3 \end{bmatrix}$$





An Inclusion Theorem for $\mathcal{R}_i(n, k), i = 0, 1, \ldots, k$

The proof follows from characterizing the maximal achievable and matching simplexes

For any k and $n > k$, we have

$$\mathcal{R}_0(n, k) \subset \mathcal{R}_1(n, k) \subset \mathcal{R}_2(n, k) \subset \cdots \subset \mathcal{R}_{k-1}(n, k) \subset \mathcal{R}_k(n, k)$$

An Inclusion Theorem for $\mathcal{R}_i(n, k), i = 0, 1, \ldots, k$

The proof follows from characterizing the maximal achievable and matching simplexes

For any k and $n > k$, we have

$$\mathcal{R}_0(n, k) \subset \mathcal{R}_1(n, k) \subset \mathcal{R}_2(n, k) \subset \cdots \subset \mathcal{R}_{k-1}(n, k) \subset \mathcal{R}_k(n, k)$$

Can we completely characterize these polytopes?

Yes, we can, but with much more work.

Perfect Matching, Matching Bound, and Greedy Matchings

TOOLS: New & Old

A fractional matching in (V, E) is a vector $\boldsymbol{w} \in \mathbb{R}^{|E|}$ whose components (weights) w_ϵ, $\epsilon \in E$, are non-negative and $\sum_{\epsilon \ni v} w_\epsilon \leqslant 1$ for each $v \in V$.
$\implies$ the matching bound

$$|V| \geqslant \sum_{v \in V} \sum_{\epsilon \ni v} w_\epsilon = \sum_{\epsilon \in E} w_\epsilon |\epsilon|.$$

A matching that saturates the bound is called perfect.

Perfect Matching, Matching Bound, and Greedy Matchings

TOOLS: New & Old

A fractional matching in (V, E) is a vector $\boldsymbol{w} \in \mathbb{R}^{|E|}$ whose components (weights) w_e, $e \in E$, are non-negative and $\sum_{e \ni v} w_e \leqslant 1$ for each $v \in V$.
$\implies$ the matching bound

$$|V| \geqslant \sum_{v \in V} \sum_{e \ni v} w_e = \sum_{e \in E} w_e |e|.$$

A matching that saturates the bound is called perfect.

Greedy Matching Theorem

Let $\boldsymbol{\lambda}$ be a point in $\mathcal{R}_i(n, k)$, i.e., $\exists \boldsymbol{w} \in \mathrm{FMP}(\Gamma_i(n, k))$ s.t. $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{\boldsymbol{w}}$.

Then there exist a greedy matching $\boldsymbol{\sigma} \in \mathrm{FMP}(\Gamma_i(n, k))$ s.t. $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{\boldsymbol{\sigma}}$ and the weight of the j-th systematic edge $\sigma_j = \min\{1, \lambda_j\}$, for all $j \leqslant i$.

$\mathcal{R}_i(n,k)$ for $n \geqslant 2k$

Slicing $\Gamma_i(n,k)$, $n - i \geqslant k$, into k-Uniform Subgraphs

Consider $\boldsymbol{\lambda} = (\underbrace{\lambda_1, \ldots, \lambda_{i_A}}_{\geqslant 1}, \underbrace{\lambda_{i_A+1}, \ldots, \lambda_i, \ldots, \lambda_k}_{<1}) \in \mathcal{R}_i(n,k)$

+ matching constraint at nodes, greedy matching, $\Gamma_i(n,k)$ structure.

$\implies$ There is a perfect matching that saturates the matching bound

$$k \cdot \sum_{j=1}^{i_A} (\lambda_j - 1) + \sum_{j=i_A+1}^{i} \lambda_j + k \cdot \sum_{j=i+1}^{k} \lambda_j \leqslant n - i_A$$

$\mathcal{R}_i(n,k)$ for $n \geqslant 2k$

Slicing $\Gamma_i(n,k)$, $n - i \geqslant k$, into k-Uniform Subgraphs

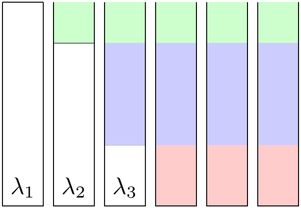
Consider $\boldsymbol{\lambda} = (\underbrace{\lambda_1, \dots, \lambda_{i_A}}_{\geqslant 1}, \underbrace{\lambda_{i_A+1}, \dots, \lambda_i, \dots, \lambda_k}_{<1}) \in \mathcal{R}_i(n,k)$

$+$ matching constraint at nodes, greedy matching, $\Gamma_i(n,k)$ structure.

$\implies$ There is a perfect matching that saturates the matching bound

$$k \cdot \sum_{j=1}^{i_A} (\lambda_j - 1) + \sum_{j=i_A+1}^{i} \lambda_j + k \cdot \sum_{j=i+1}^{k} \lambda_j \leqslant n - i_A$$

Example: A perfect matching on $\Gamma_3(6,3)$ with $i_A = 1$, λ_2=0.8, $\lambda_3 = 0.3$.



$\mathcal{R}_i(n,k)$ for $n \geqslant 2k$

Observe that when $n \geqslant 2k$, then $n - i \geqslant k$ for all $i = 0, 1, 2, \ldots, k$.

Consider $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k) \in \mathcal{R}_i(n,k)$ and a partition of $\{1, \ldots, k\}$ into sets A, B, C s.t. if $j > i$, $j \in C$; if $j \leqslant i$, $j \in A$ if $\lambda_j \geqslant 1$, otherwise $j \in B$.

$\Longrightarrow$

$\mathcal{R}_i(n,k)$ is the intersection for the following half spaces:

$\boldsymbol{\lambda} \geqslant \mathbf{0}$ and

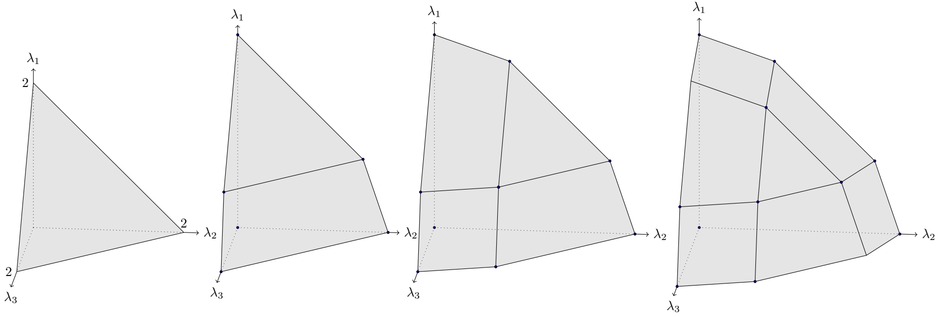
$$k \cdot \sum_{j \in A} (\lambda_j - 1) + \sum_{j \in B} \lambda_j + k \cdot \sum_{j \in C} \lambda_j \leqslant n - |A|,$$

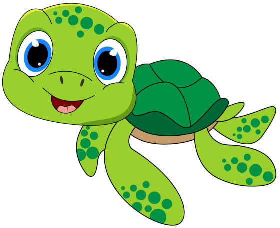
for all partitions $A \cup B = \{1, \ldots, i\}$ s.t. $\lambda_j \geqslant 1$ for $j \in A$ and $\lambda_j < 1$ for $j \in B$, and $C = \{i+1, \ldots, k\}$. If $i = k$, then $A = \emptyset$ does not give an active constraint.

$\Longrightarrow$

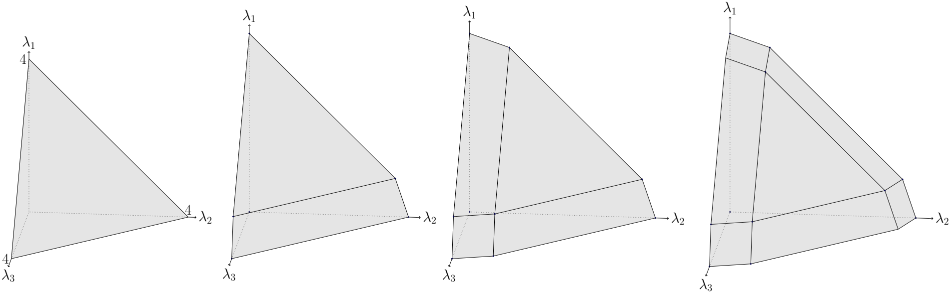
There are $k + 2^i$ hyperplanes for $i = 0, \ldots, k-1$, and $k + 2^k - 1$ for $i = k$.

$\mathcal{R}_i(6,3)$, $i = 0, 1, 2, 3$





$\mathcal{R}_i(12, 3)$, $i = 0, 1, 2, 3$



$\mathcal{R}_i(n,k)$ for $n - i < k$

$\Longrightarrow$ There may not be a matching that saturates the matching bound

$$k \cdot \sum_{j=1}^{i_A} (\lambda_j - 1) + \sum_{j=i_A+1}^{i} \lambda_j + k \cdot \sum_{j=i+1}^{k} \lambda_j \leqslant n - i_A$$

What about the vertex cover bound $\sum_{j=1}^{k} \lambda_j \leqslant i$?

$\mathcal{R}_i(n, k)$ for $n - i < k$

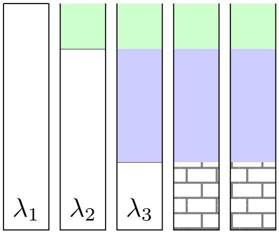
$\implies$ There may not be a matching that saturates the matching bound

$$k \cdot \sum_{j=1}^{i_A} (\lambda_j - 1) + \sum_{j=i_A+1}^{i} \lambda_j + k \cdot \sum_{j=i+1}^{k} \lambda_j \leqslant n - i_A$$

What about the vertex cover bound $\sum_{j=1}^{k} \lambda_j \leqslant i$?

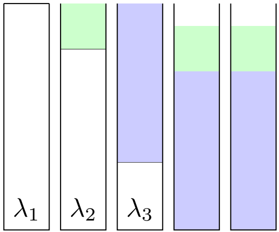
Example: Two matchings on $\Gamma_3(5, 3)$ with $i_A = 1$, λ_2=0.8, $\lambda_3 = 0.3$.

size $1 + \frac{4}{3} + \frac{2}{3}\lambda_2$



hits a brick wall

size 3



saturates the vertex cover bound

The Vertex Cover Bound on $\mathcal{R}_k(n, k)$ for $n < 2k$

Multiple points in $\mathcal{R}_k(n, k)$ saturate the bound when $n = 2k - 1$

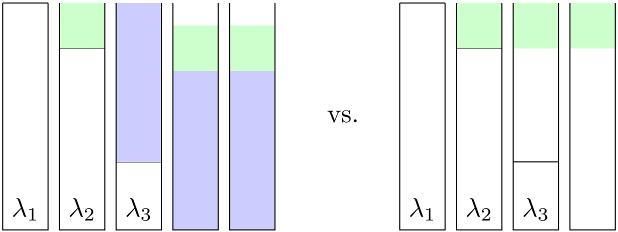
$\boldsymbol{\lambda} = (1, \ldots, 1)$ is the only point reaching the bound when $n < 2k - 1$.

The Vertex Cover Bound on $\mathcal{R}_k(n,k)$ for $n < 2k$

Multiple points in $\mathcal{R}_k(n,k)$ saturate the bound when $n = 2k-1$

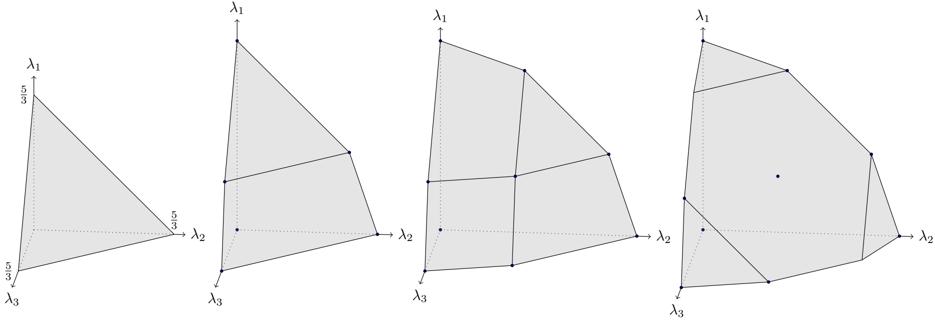
$\boldsymbol{\lambda} = (1,\ldots,1)$ is the only point reaching the bound when $n < 2k-1$.

Example: $\Gamma_3(5,3)$ vs. $\Gamma_3(4,3)$ when $i_A = 1$, λ_2=0.8, $\lambda_3 = 0.3$.



$\mathcal{R}_i(5,3)$, $i = 0,1,2,3$

and some remarks on $\mathcal{R}_i(2k-1,k)$

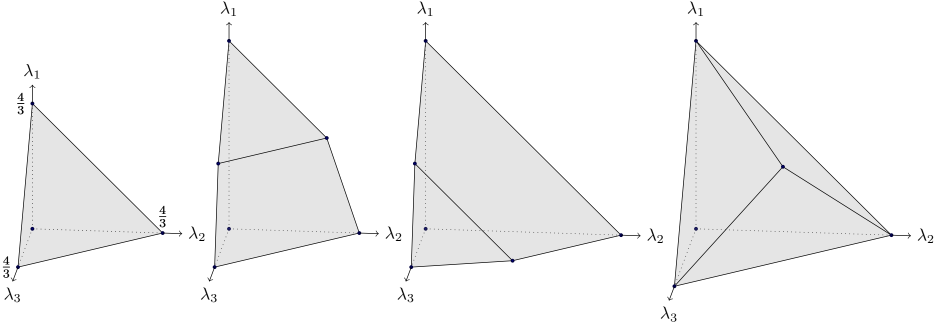


In $\mathcal{R}_i(2k-1,k)$,

- if $i < k$, then $n - i \geqslant k$ for which we know $n + 2^i$ matching bounds.
- if $i = k$, multiple points achieve the vertex cover bound.

$\mathcal{R}_i(4,3)$, $i = 0, 1, 2, 3$

and some remarks on $\mathcal{R}_i(n,k)$ for $n < 2k-1$



In $\mathcal{R}_i(n,k)$ for $n < 2k-1$,

- if $n - i \geqslant k$ and we have $n + 2^i$ matching bounds.
- if $i = k$, a single point achieves the vertex cover bound.

Service Rate Region Problem(s) Formulation

System Model:

- k data objects are stored redundantly across n nodes.
- Data objects are represented as elements of some finite field.
- Each server stores a linear combination of data objects, i.e., a coded object of the same size (same field).
- Requests for object i, $i \in \{1, \ldots, k\}$ arrive to the system at rate λ_i.
- At each node, requests are serviced at rate $\mu = 1$.

SOME OBJECTIVES:

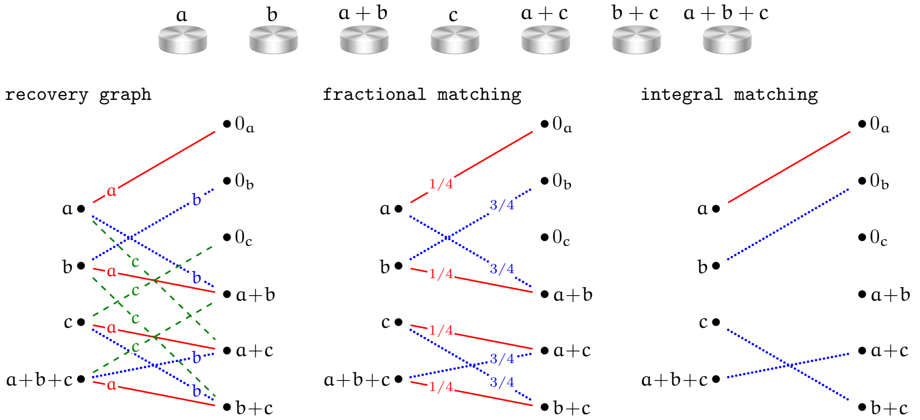
1. Determine the set of rates $(\lambda_1, \ldots, \lambda_k)$  that can be supported by the system implementing some common redundancy scheme.
2. Design a redundancy scheme in order to maximize and/or shape the of region of supported arrival rates under some limited resources.
3. Evaluate the system's performance for a given stochastic model of $(\lambda_1, \ldots, \lambda_k)$ (e.g., probability of supported rates, load imbalance).

Asynchronism

Asynchronous Service Rate Region

Asynchronous Batch Codes by Riet, Skachek, and Thomas

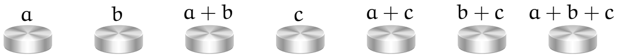
Consider the $(7,3)$ simplex code and two ways to satisfy demand $(1,3,0)$:

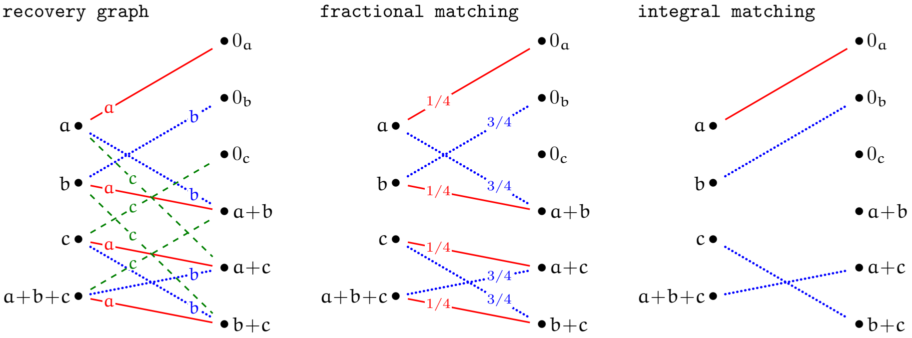


Asynchronous Service Rate Region

Asynchronous Batch Codes by Riet, Skachek, and Thomas

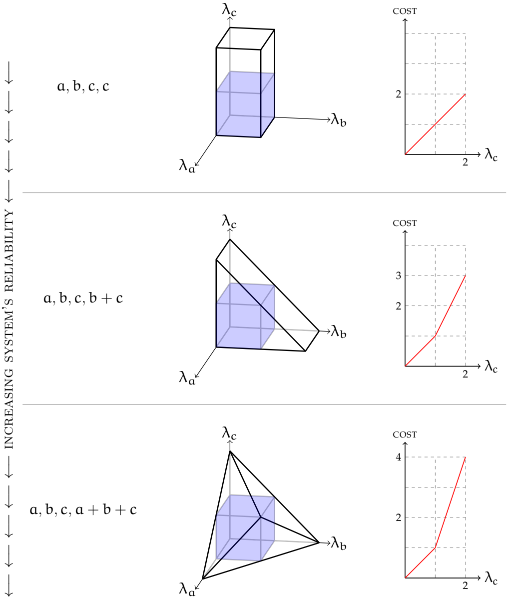
Consider the $(7,3)$ simplex code and two ways to satisfy demand $(1,3,0)$:





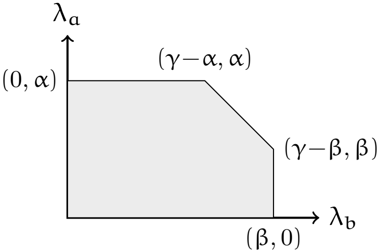
Q: If some users leave the system, can others use the freed resources?

Benefits and Costs of Adding Server(s)



Covering a Region with Minimal Storage

We need to serve requests in the region $\lambda_a \leqslant \alpha$, $\lambda_b \leqslant \beta$, $\lambda_a + \lambda_b \leqslant \gamma$.



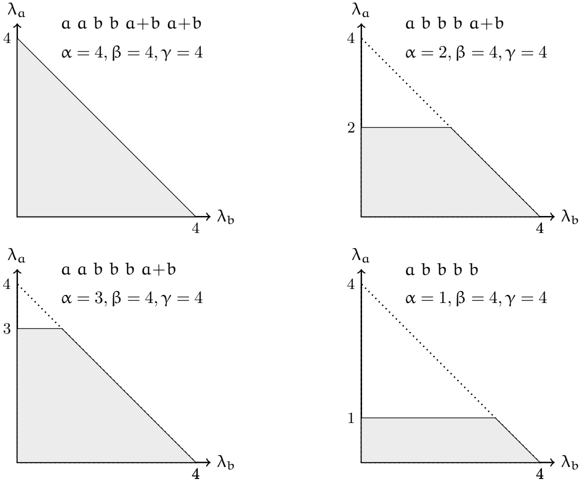
The columns of the generator matrix can only be $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$.

$$\underbrace{\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\ldots\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]}_{n_{\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]}}\quad\underbrace{\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\ldots\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]}_{n_{\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]}}\quad\underbrace{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]\ldots\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]}_{n_{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]}}$$

Find $n_{\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]}, n_{\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]}, n_{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]}$ that minimize $n = n_{\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]} + n_{\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]} + n_{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]}$.

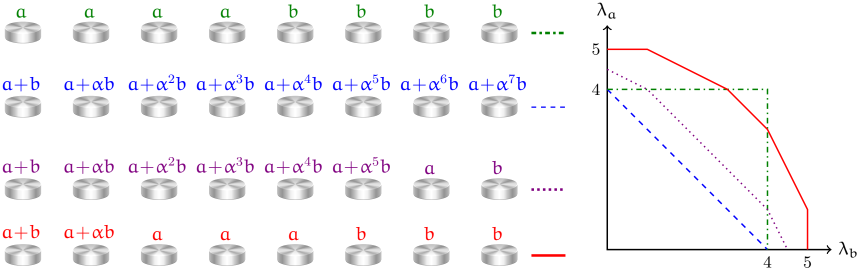
Covering a Region with Minimal Storage – Examples

What is the minimal number of servers and the redundancy scheme that satisfy the demand described by $\lambda_a \leqslant \alpha$, $\lambda_b \leqslant \beta$, $\lambda_a + \lambda_b \leqslant \gamma$?



Maximizing Service Rate Region with Fixed Resources

How should we store k objects on n servers?



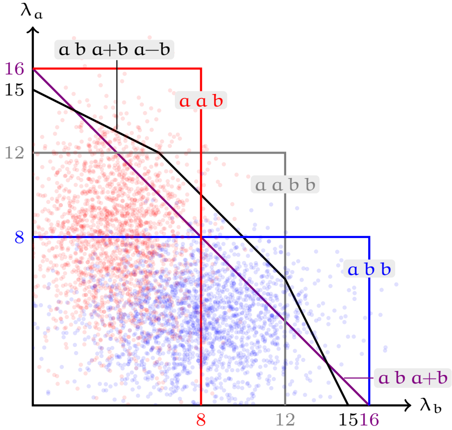
- Combining coding and replication is beneficial in multiple ways.
- Service rate region depends on the generator matrix of the code.

Covered Requests, Server Utilization, Load (Im)balance

Requests: $\lambda_a \sim \mathcal{N}^+(4,4)$ and $\lambda_b \sim \mathcal{N}^+(8,8)$ and vice versa.
Two systems with equal total service bandwidth, storing $k = 2$ objects.

System 1: $n = 3$ with $\mu = 8$ with codes
$[a, a, b]$ $[a, b, b]$ $[a, b, a{+}b]$

System 2: $n = 4$ with $\mu = 6$ with codes
$[a, a, b, b]$ $[a, b, a{+}b, a{-}b]$



Request coverage: 0.7366 for $[a, a, b]$ & $[a, b, b]$, 0.8727 for $[a, b, a{+}b]$
0.9211 for $[a, a, b, b]$, and 0.9434 $[a, b, a{+}b, a{-}b]$.

Codes for (Un)Expected Loads

New applications create new performance metrics for codes, and thus the needs for new coding schemes to be designed.