Mixed Hypergraphs for Linear-Time Construction of Denser Hashing-Based Data Structures

Michael Rink

Ilmenau University of Technology

RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)

RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:

RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$

RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$

x_i:	Kr	ak	on	oš	
v_i:	2	1	5	5	$\in(\mathbb{Z}_6,+)$

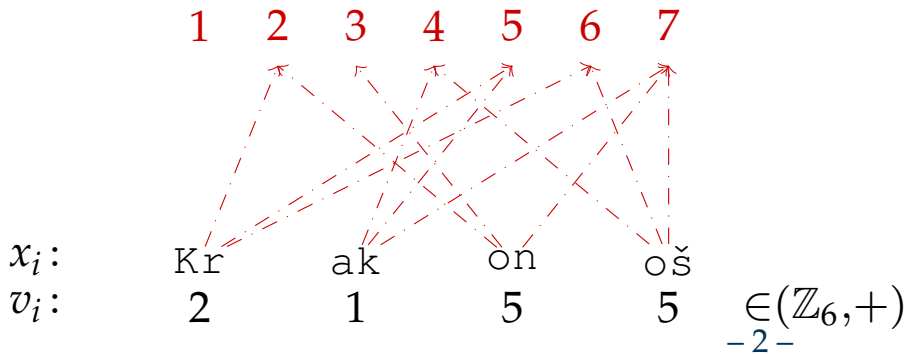
RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$
- use hash functions $h_1, h_2, \ldots, h_k$ to map each x_i to k *random* integers $h_j(x_i) \in \{1, 2, \ldots, n\}$, $j = 1, 2, \ldots, k$



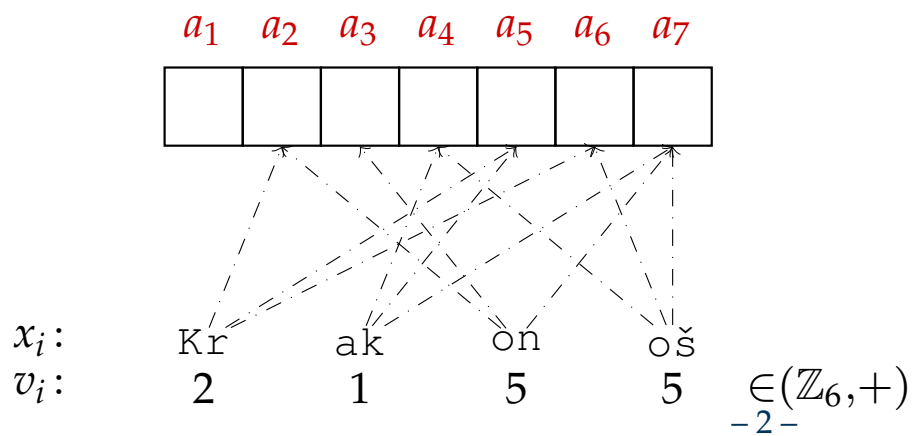
RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$
- use hash functions $h_1, h_2, \ldots, h_k$ to map each x_i to k *random* integers $h_j(x_i) \in \{1, 2, \ldots, n\}$, $j = 1, 2, \ldots, k$
- find $a_1, a_2, \ldots, a_n \in V$ s.t.



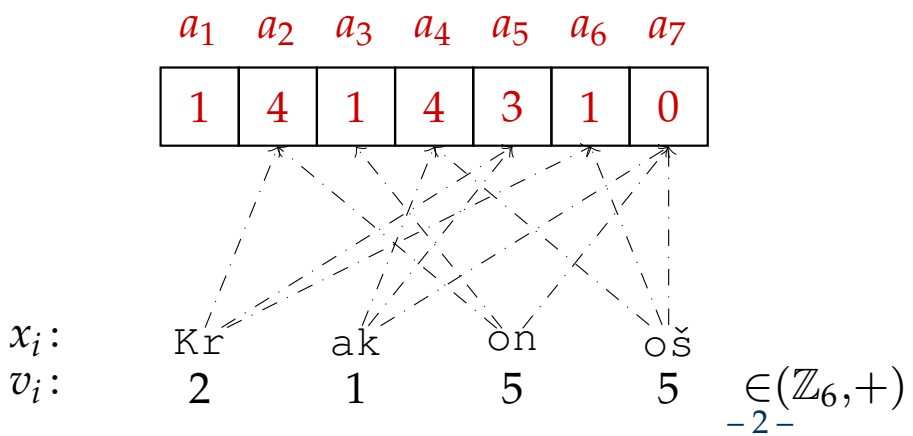
RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$
- use hash functions $h_1, h_2, \ldots, h_k$ to map each x_i to k *random* integers $h_j(x_i) \in \{1, 2, \ldots, n\}$, $j = 1, 2, \ldots, k$
- find $a_1, a_2, \ldots, a_n \in V$ s.t. $\bigoplus_j a_{h_j(x_i)} = v_i$ for $i = 1, 2, \ldots, m$



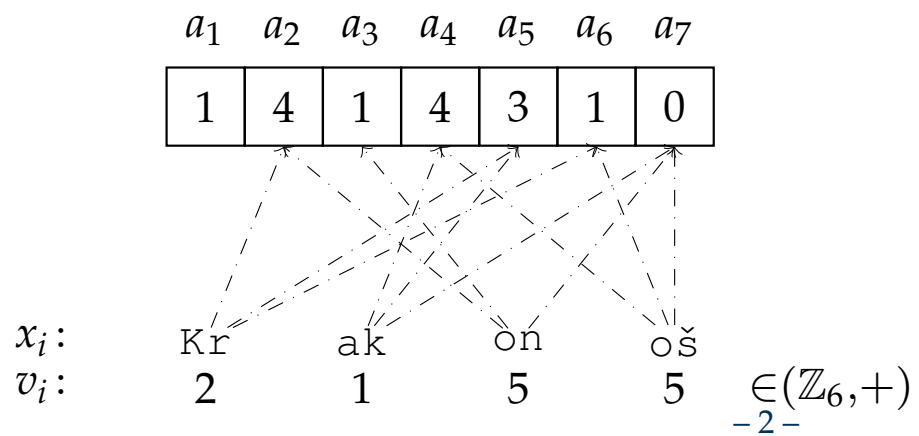
RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$
- use hash functions $h_1, h_2, \ldots, h_k$ to map each x_i to k *random* integers $h_j(x_i) \in \{1, 2, \ldots, n\}$, $j = 1, 2, \ldots, k$
- find $a_1, a_2, \ldots, a_n \in V$ s.t. $\bigoplus_j a_{h_j(x_i)} = v_i$ for $i = 1, 2, \ldots, m$
- store $a_1, a_2, \ldots, a_n$ and $h_1, h_2, \ldots, h_k$;



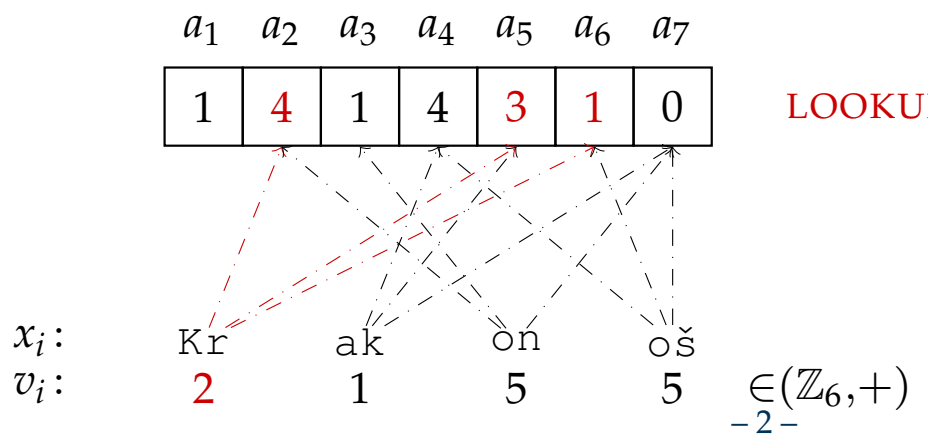
RETRIEVAL DATA STRUCTURE

Problem:

- given m key-value pairs (x_i, v_i), $i = 1, \ldots, m$ (distinct keys)
- build data structure that on LOOKUP(x) returns:
  v_i if there is an x_i with $x_i = x$; otherwise an arbitrary value

Construction [Chazelle et al., 2004]:

- assume that values v_i are from abelian group $(V, \oplus)$
- use hash functions $h_1, h_2, \ldots, h_k$ to map each x_i to k *random* integers $h_j(x_i) \in \{1, 2, \ldots, n\}$, $j = 1, 2, \ldots, k$
- find $a_1, a_2, \ldots, a_n \in V$ s.t. $\bigoplus_j a_{h_j(x_i)} = v_i$ for $i = 1, 2, \ldots, m$
- store $a_1, a_2, \ldots, a_n$ and $h_1, h_2, \ldots, h_k$; LOOKUP$(x) := \bigoplus_j a_{h_j(x)}$



2-CORE

We would like to *solve* the system of equations in *in linear time*.

2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$:

2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$

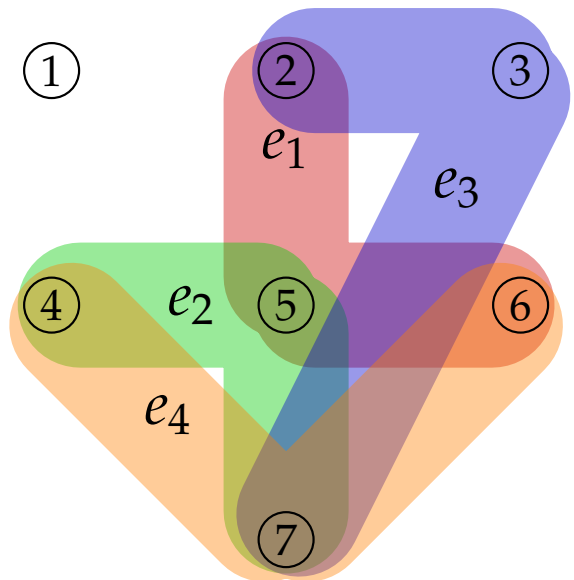
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



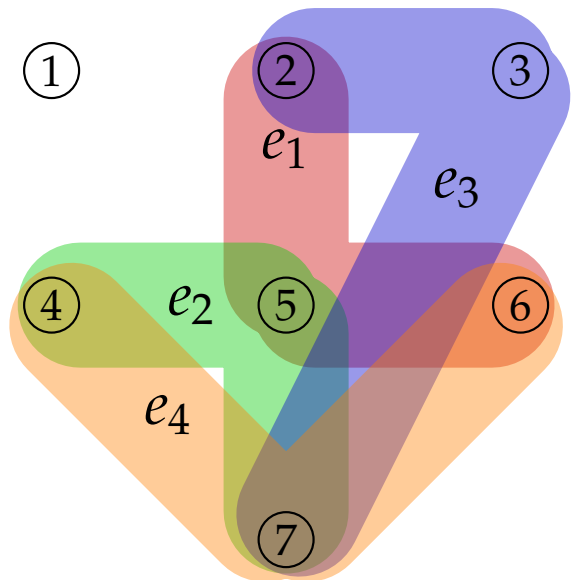
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2

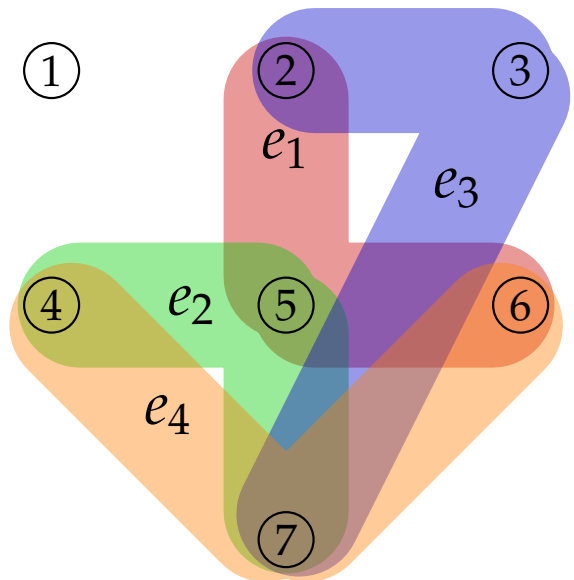
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm

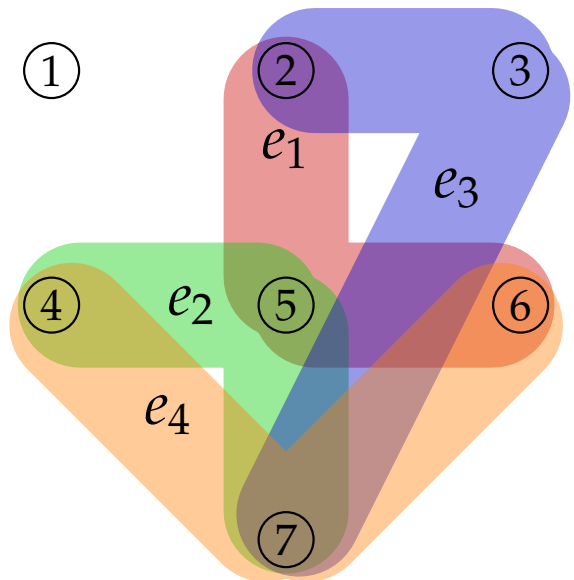
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

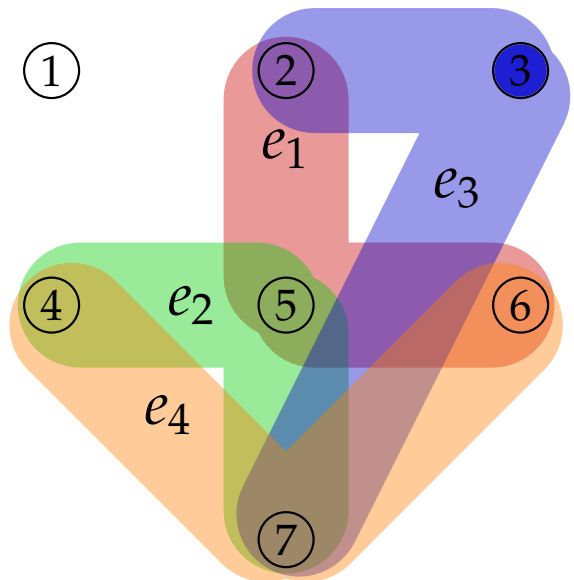
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	1	2	3	4	5	6	7
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_3$	0	1	1	0	0	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

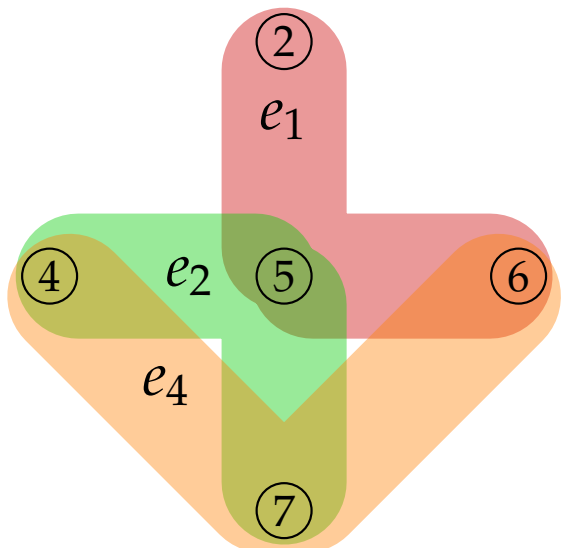
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	1	4	5	6	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

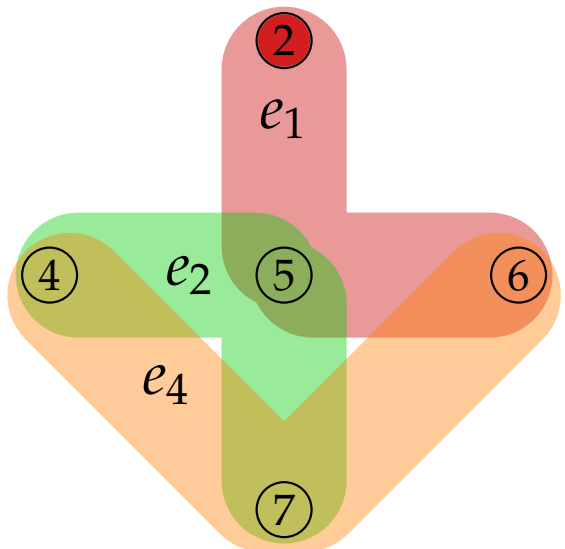
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	1	4	5	6	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

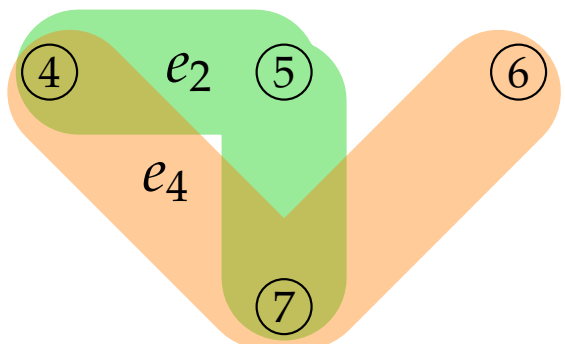
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	1	4	5	6	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

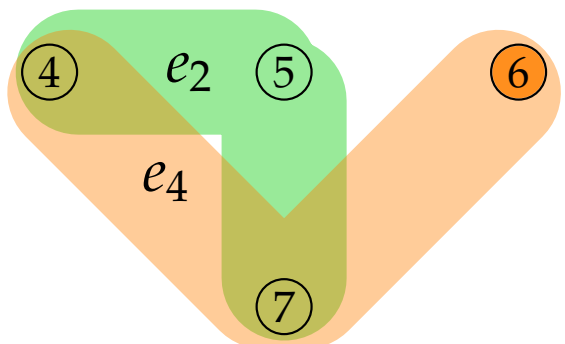
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	1	4	5	6	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	0	0	1	1	0
$\boldsymbol{b}_2$	0	0	0	1	1	0	1
$\boldsymbol{b}_4$	0	0	0	1	0	1	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

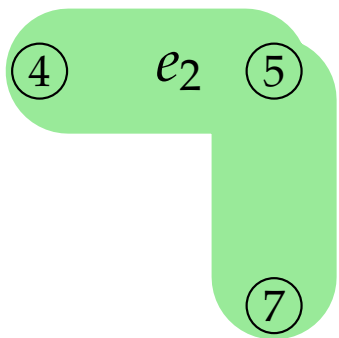
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	6	4	5	1	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	1	0	1	0	0
$\boldsymbol{b}_4$	0	0	1	1	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

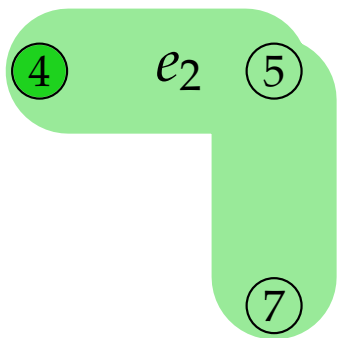
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	6	4	5	1	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	1	0	1	0	0
$\boldsymbol{b}_4$	0	0	1	1	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$

	3	2	6	4	5	1	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	1	0	1	0	0
$\boldsymbol{b}_4$	0	0	1	1	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1

2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

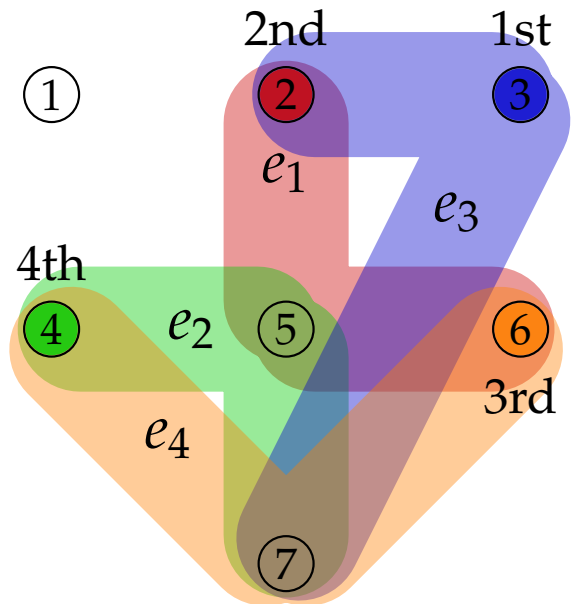
2-CORE

We would like to *solve* the system of equations in *in linear time*.

Matrix $\boldsymbol{M}$: row i is the characteristic vector $\boldsymbol{b}_i$ of $\{h_1(x_i), h_2(x_i), \ldots, h_k(x_i)\}$

	3	2	6	4	5	1	7
$\boldsymbol{b}_3$	1	1	0	0	0	0	1
$\boldsymbol{b}_1$	0	1	1	0	1	0	0
$\boldsymbol{b}_4$	0	0	1	1	0	0	1
$\boldsymbol{b}_2$	0	0	0	1	1	0	1

Hypergraph $\mathcal{H}$: incidence matrix $\boldsymbol{M}$



2-Core of $\mathcal{H}$:

- largest induced sub-hypergraph with min-degree ≥ 2
- determined via "peeling" algorithm
- $\mathcal{H}$ has empty 2-core $\Leftrightarrow$ row echelon form of $\boldsymbol{M}$ solely via permutations $\Rightarrow$ can solve the system in linear time

APPEARANCE OF THE 2-CORE (1)

Model — k-uniform hypergraph:

- n nodes
- m *random* edges of size k from $\binom{\{1,2,\ldots,n\}}{k}$

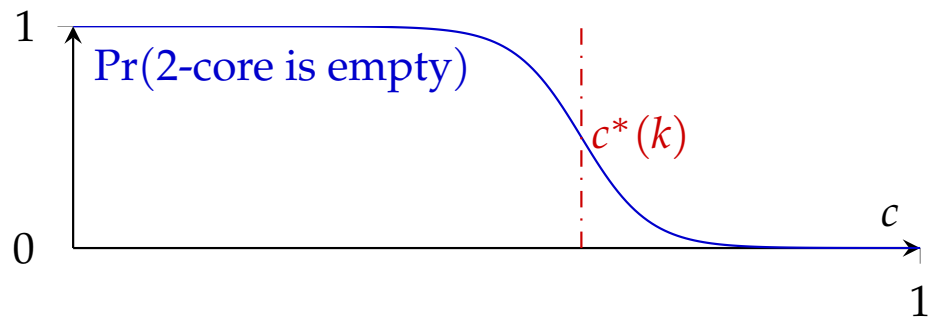
APPEARANCE OF THE 2-CORE (1)

Model — k-uniform hypergraph:

- n nodes
- m *random* edges of size k from $\binom{\{1,2,\ldots,n\}}{k}$

Well known, claim [Majewski et al., 1996], proof [Molloy, 2004]:

- Let $k \geq 3$. For increasing $c = m/n$ there is a *sharp phase transition* from empty to non-empty 2-core.



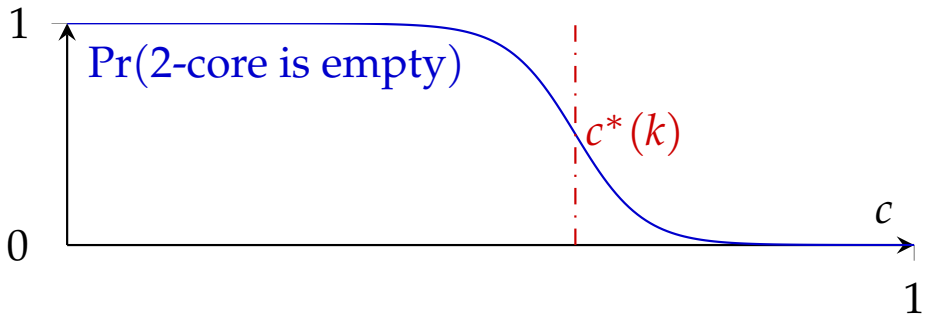
APPEARANCE OF THE 2-CORE (1)

Model — k-uniform hypergraph:

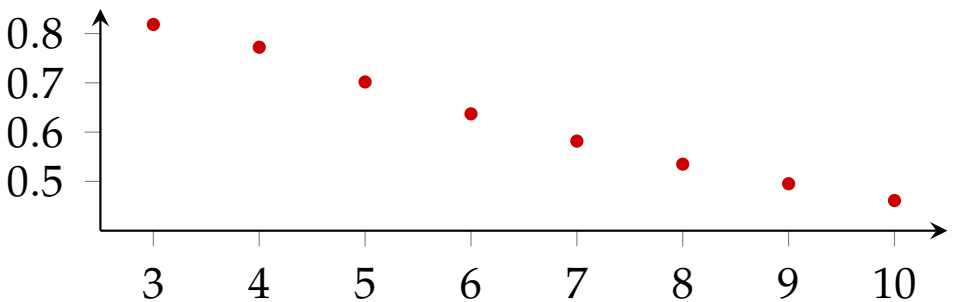
- n nodes
- m *random* edges of size k from $\binom{\{1,2,\ldots,n\}}{k}$

Well known, claim [Majewski et al., 1996], proof [Molloy, 2004]:

- Let $k \geq 3$. For increasing $c = m/n$ there is a *sharp phase transition* from empty to non-empty 2-core.



- Thresholds $c^*(k)$, maximum is $c^*(3) \approx 0.818$



APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)

APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)
- PHF [Chazelle et al., 2004, Botelho et al., 2007] (later)

APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)
- PHF [Chazelle et al., 2004, Botelho et al., 2007] (later)
- IBLT [Goodrich and Mitzenmacher, 2011]

APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)
- PHF [Chazelle et al., 2004, Botelho et al., 2007] (later)
- IBLT [Goodrich and Mitzenmacher, 2011]
- Biff Codes [Mitzenmacher and Varghese, 2012]

APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)
- PHF [Chazelle et al., 2004, Botelho et al., 2007] (later)
- IBLT [Goodrich and Mitzenmacher, 2011]
- Biff Codes [Mitzenmacher and Varghese, 2012]

Can we overcome the limit of $c^*(3)$?

APPEARANCE OF THE 2-CORE (2)

The 2-core threshold c^* limits the maximum "load" $c = m/n$, $c < c^*$, of several data structures based on hashing, such as:

- Retrieval DS [Chazelle et al., 2004] (already seen)
- PHF [Chazelle et al., 2004, Botelho et al., 2007] (later)
- IBLT [Goodrich and Mitzenmacher, 2011]
- Biff Codes [Mitzenmacher and Varghese, 2012]

Can we overcome the limit of $c^*(3)$?

Yes, via *mixing* uniform hypergraphs with *different edge sizes*!

Outline

Next . . .

Related Work

Results on 2-Cores of Mixed Hypergraphs

Application: Perfect Hash Function

Summary

BACKGROUND

Irregular hypergraphs, as used in erasure correcting codes:

- parameter(s): distribution of node degrees (and edge sizes)

BACKGROUND

Irregular hypergraphs, as used in erasure correcting codes:

- parameter(s): distribution of node degrees (and edge sizes)
- positive: threshold where 2-core appears *can be improved* compared to regular hypergraphs [Luby et al., 2001, Luby, 2002, Maymounkov, 2002, Shokrollahi, 2006],. . .

BACKGROUND

Irregular hypergraphs, as used in erasure correcting codes:

- parameter(s): distribution of node degrees (and edge sizes)
- positive: threshold where 2-core appears *can be improved* compared to regular hypergraphs [Luby et al., 2001, Luby, 2002, Maymounkov, 2002, Shokrollahi, 2006],. . .

Non-uniform (mixed) hypergraphs:

- parameter: distribution of edge sizes

Background

Irregular hypergraphs, as used in erasure correcting codes:

- parameter(s): distribution of node degrees (and edge sizes)
- positive: threshold where 2-core appears *can be improved* compared to regular hypergraphs [Luby et al., 2001, Luby, 2002, Maymounkov, 2002, Shokrollahi, 2006],. . .

Non-uniform (mixed) hypergraphs:

- parameter: distribution of edge sizes
- negative: threshold where 2-core edge density switches from < 1 to > 1 *cannot be improved* compared to uniform hypergraphs [Dietzfelbinger and Rink, 2012]

Next . . .

Related Work

Results on 2-Cores of Mixed Hypergraphs

Application: Perfect Hash Function

Summary

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$,

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$.

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$ and global minimum

$$c^*(\boldsymbol{k}, \boldsymbol{\alpha}) := \min_{z \in (0,1)} T(z; \boldsymbol{k}, \boldsymbol{\alpha}) \ ,$$

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$ and global minimum

$$c^*(\boldsymbol{k}, \boldsymbol{\alpha}) := \min_{z \in (0,1)} T(z; \boldsymbol{k}, \boldsymbol{\alpha}) \ ,$$

such that with probability $1 - o(1)$ for $m \to \infty$ the following holds:

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$ and global minimum

$$c^*(\boldsymbol{k}, \boldsymbol{\alpha}) := \min_{z \in (0,1)} T(z; \boldsymbol{k}, \boldsymbol{\alpha}) \ ,$$

such that with probability $1 - o(1)$ for $m \to \infty$ the following holds:

- *if $c < c^*(\boldsymbol{k}, \boldsymbol{\alpha})$, then $\mathcal{H}_{\text{mix}}$ has an empty 2-core.*

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$ and global minimum

$$c^*(\boldsymbol{k}, \boldsymbol{\alpha}) := \min_{z \in (0,1)} T(z; \boldsymbol{k}, \boldsymbol{\alpha}) \ ,$$

such that with probability $1 - o(1)$ for $m \to \infty$ the following holds:

- *if $c < c^*(\boldsymbol{k}, \boldsymbol{\alpha})$, then $\mathcal{H}_{\text{mix}}$ has an empty 2-core.*
- *if $c > c^*(\boldsymbol{k}, \boldsymbol{\alpha})$, then $\mathcal{H}_{\text{mix}}$ has a non-empty 2-core.*

STARTING POINT

Theorem (generalization of [Molloy, 2004])

Let $\mathcal{H}_{\text{mix}}$ be a mixture of $s \geq 1$ random k_i-uniform hypergraphs $\mathcal{H}_i$, $k_i \geq 3$, where $\mathcal{H}_i$ has $\alpha_i \cdot c \cdot n$ edges of size k_i, $i = 1, 2, \ldots, s$, and $\alpha_1 + \alpha_2 + \ldots + \alpha_s = 1$. There is a "key function" $T(z)$ with parameters $\underbrace{k_1, k_2, \ldots, k_s}_{\boldsymbol{k}}, \underbrace{\alpha_1, \alpha_2, \ldots, \alpha_s}_{\boldsymbol{\alpha}}$ and global minimum

$$c^*(\boldsymbol{k}, \boldsymbol{\alpha}) := \min_{z \in (0,1)} T(z; \boldsymbol{k}, \boldsymbol{\alpha}) \ ,$$

such that with probability $1 - o(1)$ for $m \to \infty$ the following holds:

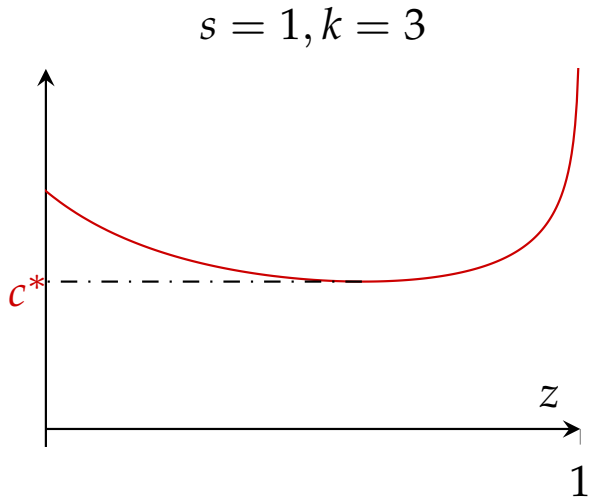
- *if $c < c^*(\boldsymbol{k}, \boldsymbol{\alpha})$, then $\mathcal{H}_{\text{mix}}$ has an empty 2-core.*
- *if $c > c^*(\boldsymbol{k}, \boldsymbol{\alpha})$, then $\mathcal{H}_{\text{mix}}$ has a non-empty 2-core.*

Proof: along the lines of [Molloy, 2004] using ideas from [Dietzfelbinger et al., 2010], see full version

KEY FUNCTION $T(z)$

Uniform (known):

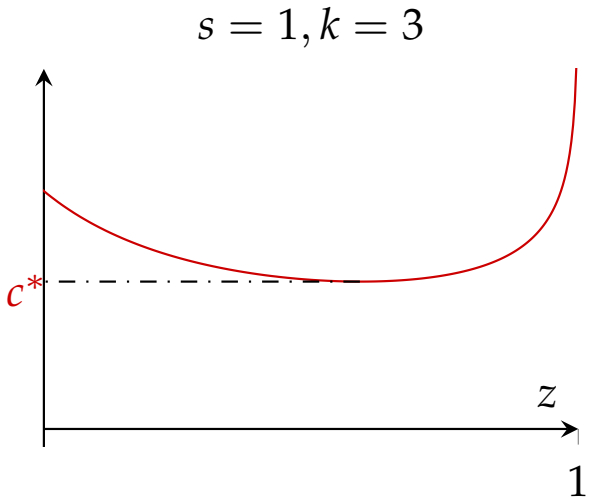
- convex, local minimum = global minimum



KEY FUNCTION $T(z)$

Uniform (known):

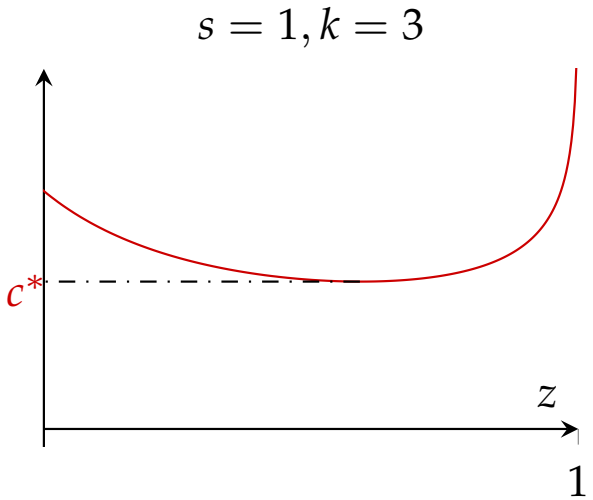
- convex, local minimum = global minimum
- function $c^*(k)$ has global maximum point at $k = 3$



KEY FUNCTION $T(z)$

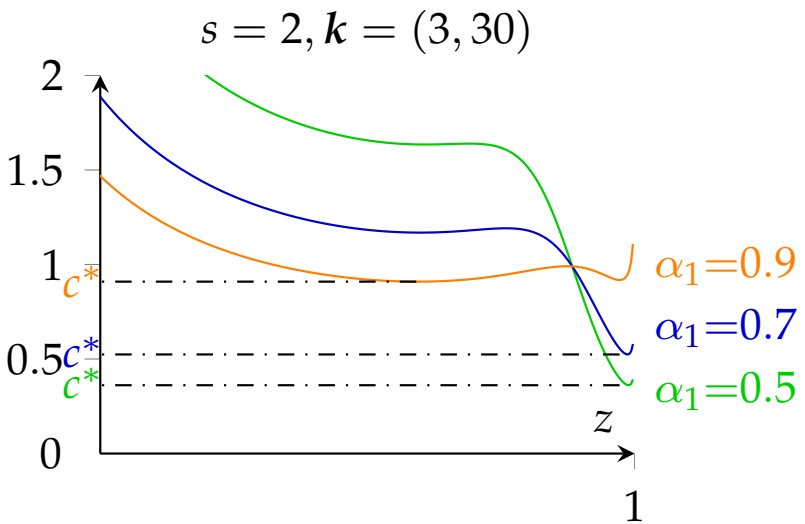
Uniform (known):

- convex, local minimum = global minimum
- function $c^*(k)$ has global maximum point at $k = 3$



Mixed (new):

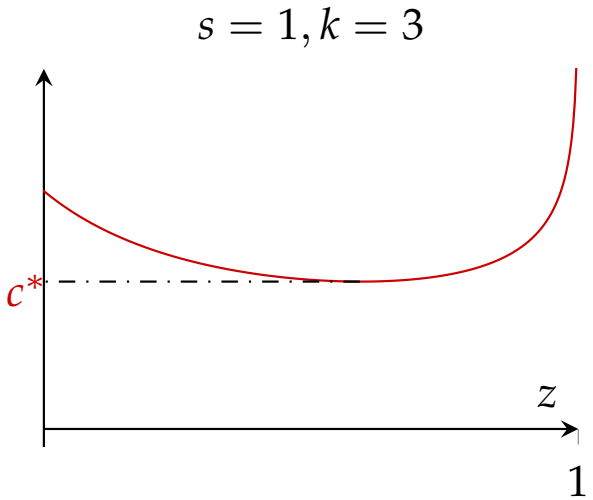
- non-convex, parameter space of dim. $2s - 1$



KEY FUNCTION $T(z)$

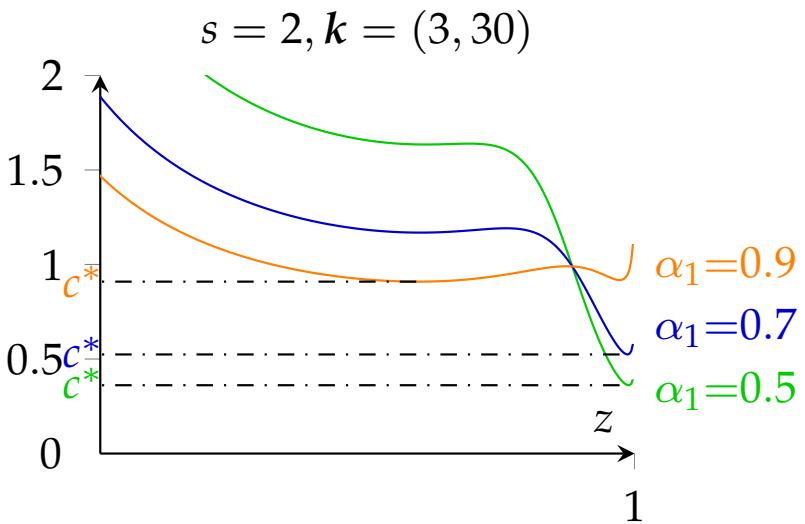
Uniform (known):

- convex, local minimum = global minimum
- function $c^*(k)$ has global maximum point at $k = 3$



Mixed (new):

- non-convex, parameter space of dim. $2s - 1$
- interested in $c^*(\boldsymbol{k}) := \max_{\boldsymbol{\alpha}} c^*(\boldsymbol{k}, \boldsymbol{\alpha})$



MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\mathbf{k} = (k_1, k_2)$ the maximum threshold $c^*(\mathbf{k})$ can be calculated efficiently,

MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\mathbf{k} = (k_1, k_2)$ the maximum threshold $c^*(\mathbf{k})$ can be calculated efficiently, and for appropriate k_1 and k_2 this value is larger than $c^*(3)$.

MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\boldsymbol{k} = (k_1, k_2)$ the maximum threshold $c^*(\boldsymbol{k})$ can be calculated efficiently, and for appropriate k_1 and k_2 this value is larger than $c^*(3)$.

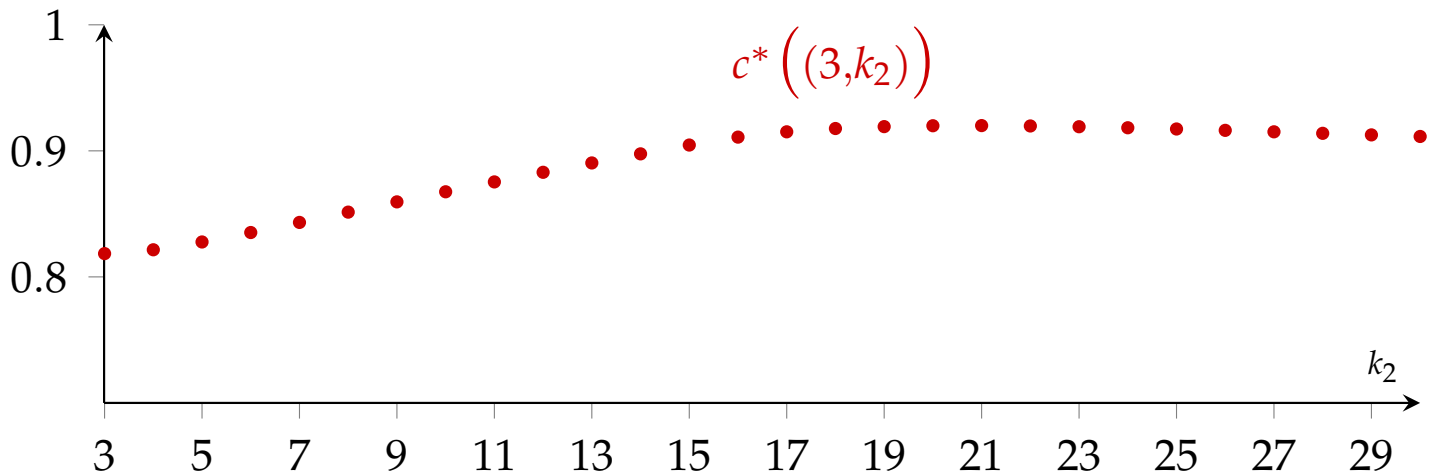
Proof: multivariate calculus, non-convex optimization, see full version

MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\mathbf{k} = (k_1, k_2)$ the maximum threshold $c^*(\mathbf{k})$ can be calculated efficiently, and for appropriate k_1 and k_2 this value is larger than $c^*(3)$.

Proof: multivariate calculus, non-convex optimization, see full version

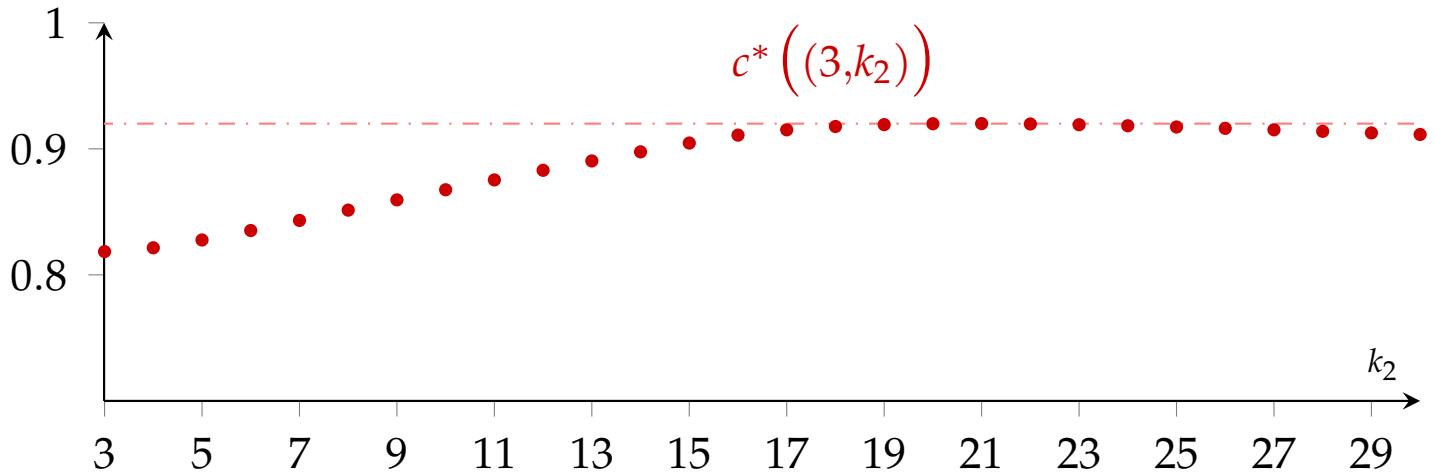


MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\boldsymbol{k} = (k_1, k_2)$ the maximum threshold $c^*(\boldsymbol{k})$ can be calculated efficiently, and for appropriate k_1 and k_2 this value is larger than $c^*(3)$.

Proof: multivariate calculus, non-convex optimization, see full version



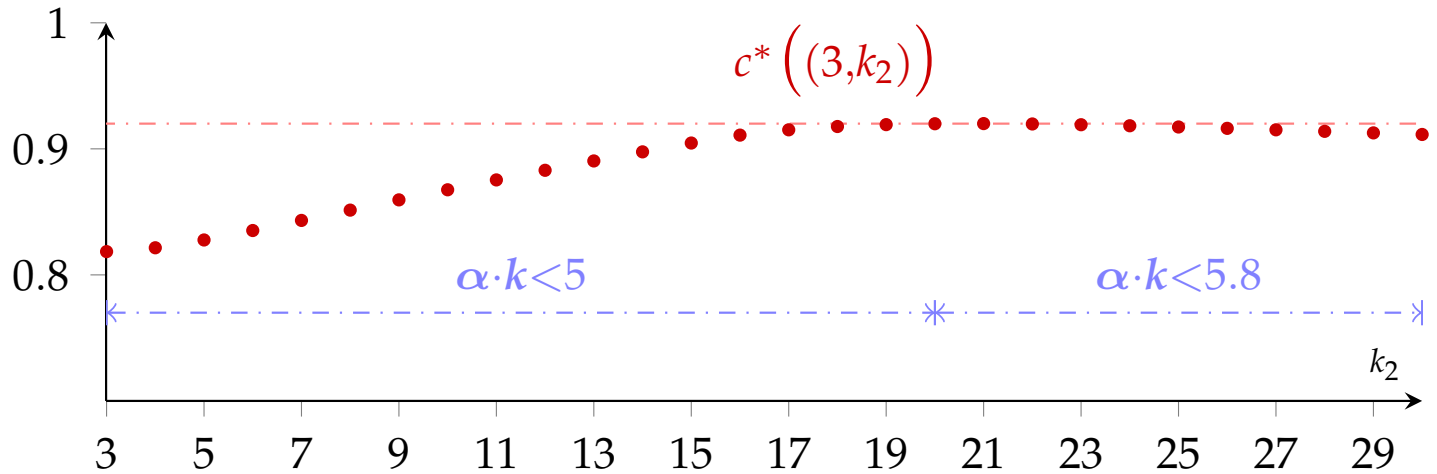
The maximum found is about 0.92 using $\boldsymbol{k} = (3, 21)$

MAXIMUM THRESHOLDS FOR MIXTURE OF TWO

Theorem

For two edge sizes $\boldsymbol{k} = (k_1, k_2)$ the maximum threshold $c^*(\boldsymbol{k})$ can be calculated efficiently, and for appropriate k_1 and k_2 this value is larger than $c^*(3)$.

Proof: multivariate calculus, non-convex optimization, see full version



The maximum found is about 0.92 using $\boldsymbol{k} = (3, 21)$ with average edge size around 5.

EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$

EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold

EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold
- for each $(n, m = c \cdot n, k_2, \boldsymbol{\alpha})$ build 100 random hypergraphs

EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold
- for each $(n, m = c \cdot n, k_2, \boldsymbol{\alpha})$ build 100 random hypergraphs
- apply "peeling" algorithm, a non-empty 2-core is considered a failure

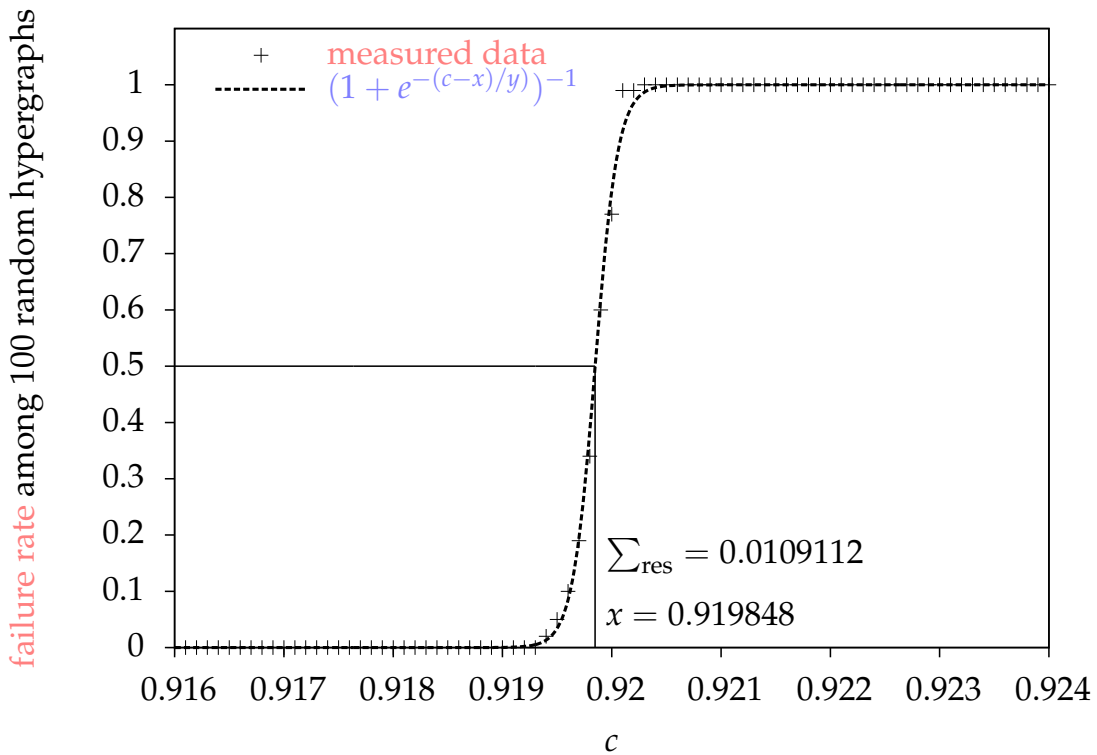
EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold
- for each $(n, m = c \cdot n, k_2, \boldsymbol{\alpha})$ build 100 random hypergraphs
- apply "peeling" algorithm, a non-empty 2-core is considered a failure

Results:

$$\boldsymbol{k} = (3, 21)$$
$$c^*(\boldsymbol{k}) = 0.920$$



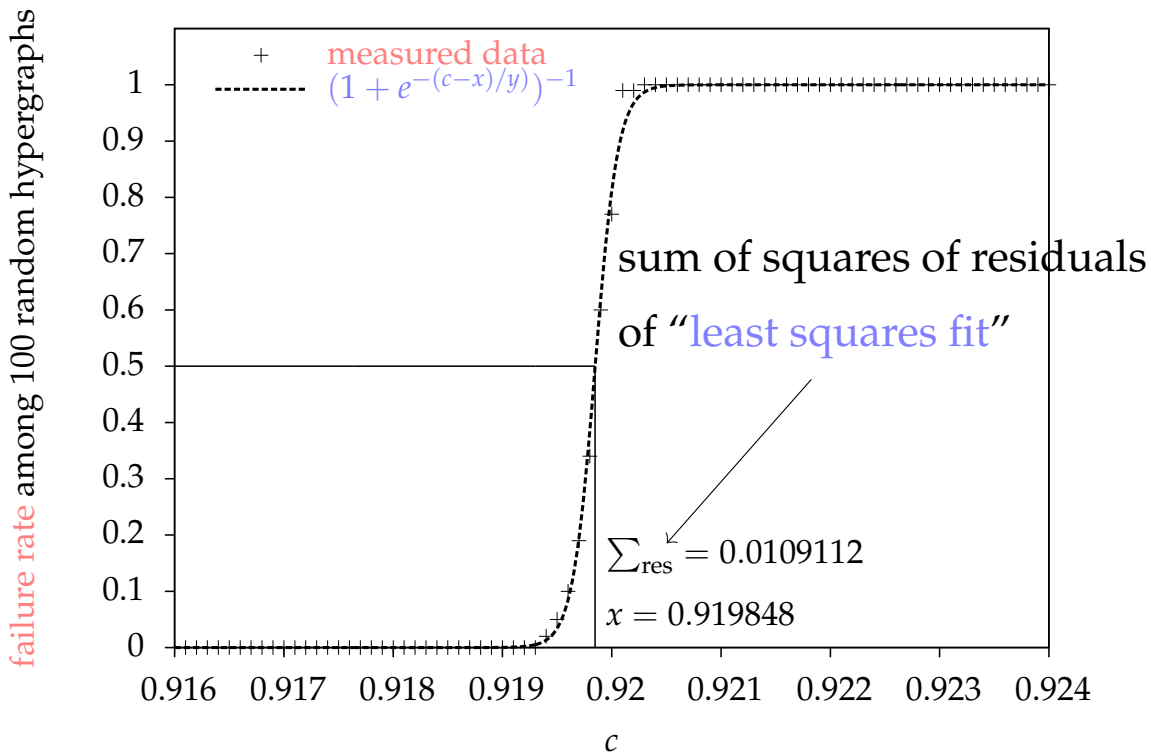
EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold
- for each $(n, m = c \cdot n, k_2, \boldsymbol{\alpha})$ build 100 random hypergraphs
- apply "peeling" algorithm, a non-empty 2-core is considered a failure

Results:

$$\boldsymbol{k} = (3, 21)$$
$$c^*(\boldsymbol{k}) = 0.920$$



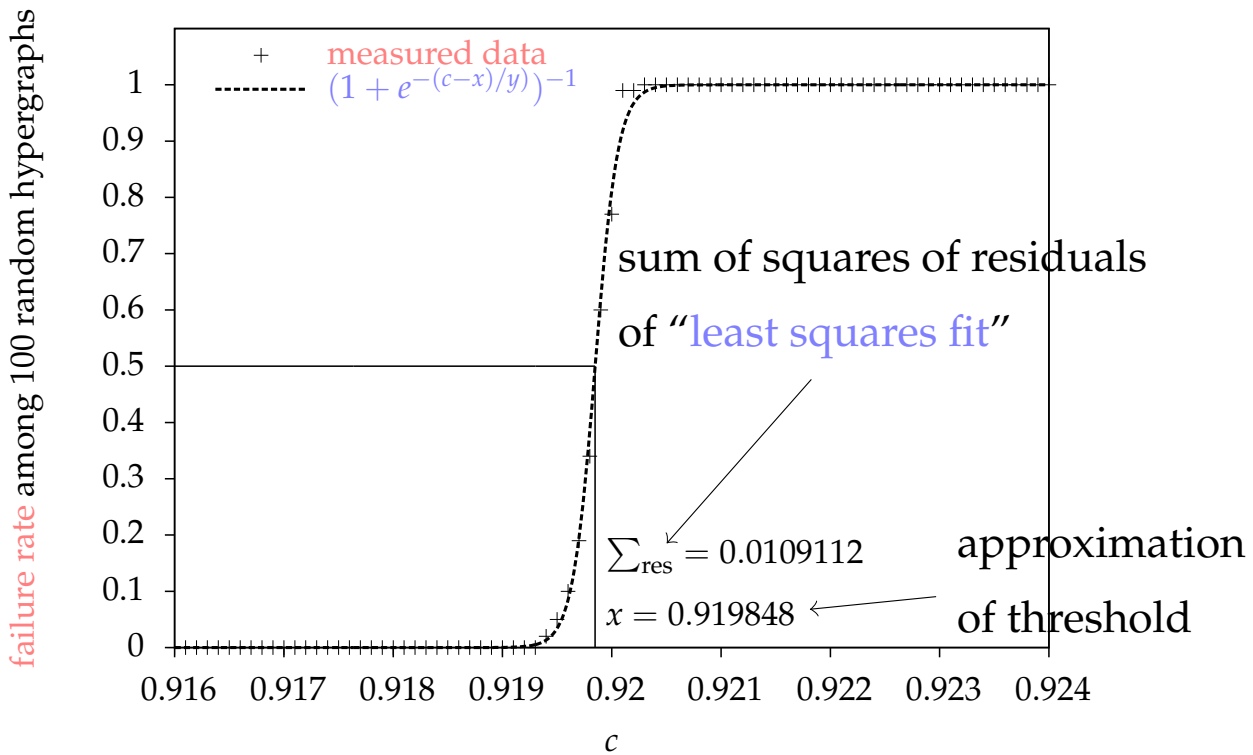
EXPERIMENTS

Setup:

- $n = 10^7$ nodes, edge sizes $\boldsymbol{k} = (3, k_2)$, optimal $\boldsymbol{\alpha}$
- edge densities c around theoretical 2-core threshold
- for each $(n, m = c \cdot n, k_2, \boldsymbol{\alpha})$ build 100 random hypergraphs
- apply "peeling" algorithm, a non-empty 2-core is considered a failure

Results:

$$\boldsymbol{k} = (3, 21)$$
$$c^*(\boldsymbol{k}) = 0.920$$



Next . . .

HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r,

HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U

HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$

High Level Description

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that

HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that $g_{v_i}(x_i)$, $i = 1, 2, \ldots, m$, are pairwise distinct

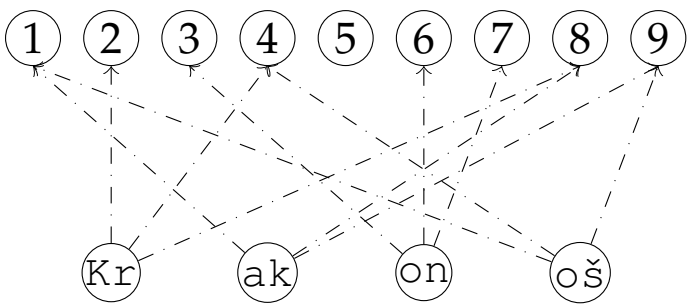
HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that $g_{v_i}(x_i)$, $i = 1, 2, \ldots, m$, are pairwise distinct



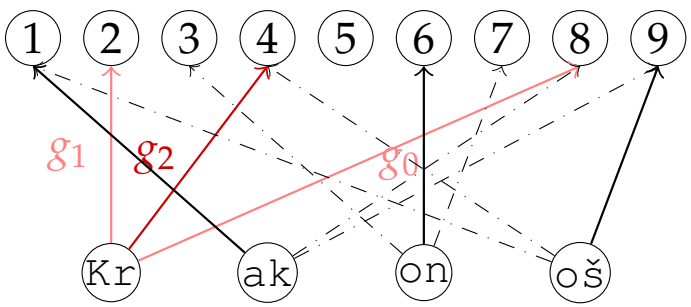
HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that $g_{v_i}(x_i)$, $i = 1, 2, \ldots, m$, are pairwise distinct



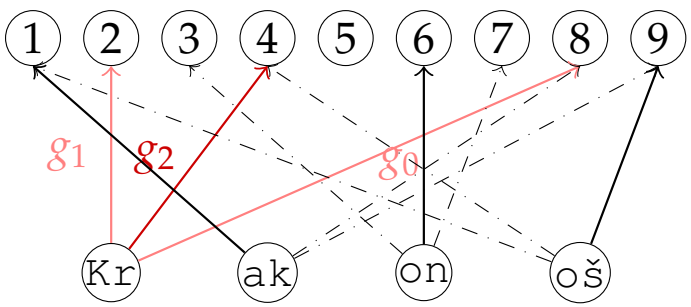
HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that $g_{v_i}(x_i)$, $i = 1, 2, \ldots, m$, are pairwise distinct



- retrieval: build retrieval data structure for the key-index pairs (x_i, v_i), $i = 1, 2, \ldots, m$, as seen before

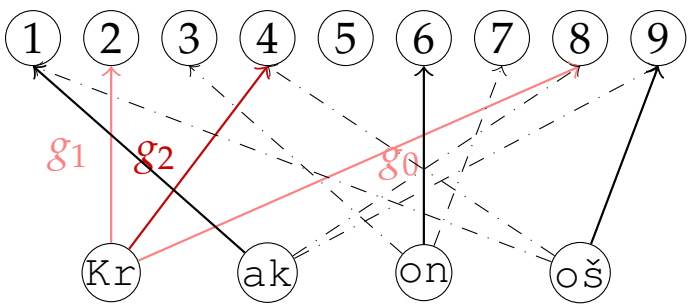
HIGH LEVEL DESCRIPTION

Problem:

- given set of m keys $S = \{x_1, x_2, \ldots, x_m\}$ and parameter r, usually S is subset of a larger universe U
- build data structure that represents an arbitrary function $f : U \to \{1, 2, \ldots, r\}$, *injective* on S

Construction, extends [Chazelle et al., 2004, Botelho et al., 2007]:

- matching: given $g_0, g_1, \ldots, g_{d-1} : U \to \{1, 2, \ldots, r\}$, for each x_i determine index v_i such that $g_{v_i}(x_i)$, $i = 1, 2, \ldots, m$, are pairwise distinct



- retrieval: build retrieval data structure for the key-index pairs (x_i, v_i), $i = 1, 2, \ldots, m$, as seen before
- define $f(x) := g_v(x)$, with $v = \text{LOOKUP}(x)$

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g.
  [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp
- construction time (expected): $O(m)$

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp
- construction time (expected): $O(m)$

Overall:

- range of PHF: $1.1 \cdot m$

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp
- construction time (expected): $O(m)$

Overall:

- range of PHF: $1.1 \cdot m$
- space usage (asympt.): $1.1 \cdot 8/5 = 1.76$ bits per key

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp
- construction time (expected): $O(m)$

Overall:

- range of PHF: $1.1 \cdot m$
- space usage (asympt.): $1.1 \cdot 8/5 = 1.76$ bits per key
- cell probes for evaluation: average 5, worst-case 16

SUITABLE PARAMETERS FOR PERFECT HASHING

Matching (known):

- for $d = 3$ and $r = 1.1 \cdot m$ matching exists whp, e.g. [Frieze and Melsted, 2009, Fountoulakis and Panagiotou, 2010]
- constr. time (expected): $O(m)$, conj. [Dietzfelbinger et al., 2010]

Retrieval (improved):

- for $k_1 = 3$, $k_2 = 16$, and $n = 1.1 \cdot m$ the 2-core is empty whp
- construction time (expected): $O(m)$

Overall:

- range of PHF: $1.1 \cdot m$
- space usage (asympt.): $1.1 \cdot 8/5 = 1.76$ bits per key
- cell probes for evaluation: average 5, worst-case 16
- construction time (expected): $O(m)$

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

Theoretical space usage is 1.76 bits per key, for $m \to \infty$.

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

Theoretical space usage is 1.76 bits per key, for $m \to \infty$.

Results:

Intel Xeon E5450 3GHz					
m	10^3	10^4	10^5	10^6	10^7
avg. time in sec					
avg. bits per key					

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

Theoretical space usage is 1.76 bits per key, for $m \to \infty$.

Results:

Intel Xeon E5450 3GHz					
m	10^3	10^4	10^5	10^6	10^7
avg. time in sec	0.11	0.32	1.51	10.63	146.18
avg. bits per key					

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

Theoretical space usage is 1.76 bits per key, for $m \to \infty$.

Results:

Intel Xeon E5450 3GHz					
m	10^3	10^4	10^5	10^6	10^7
avg. time in sec	0.11	0.32	1.51	10.63	146.18
avg. bits per key	11.62	2.86	1.89	1.78	1.76

EXPERIMENTS (NOT IN PAPER)

Setup:

- keys: about 10^7 `strings`, titles of articles and labels for categories of Wikipedia (en)
- hash functions: simple linear functions with random coefficients
- build PHFs with range $1.1 \cdot m$ for key sets of size m, repeat $10^8/m$ times

Theoretical space usage is 1.76 bits per key, for $m \to \infty$.

Results:

Intel Xeon E5450 3GHz					
m	10^3	10^4	10^5	10^6	10^7
avg. time in sec	0.11	0.32	1.51	10.63	146.18
avg. bits per key	11.62	2.86	1.89	1.78	1.76

Next . . .

Related Work

Results on 2-Cores of Mixed Hypergraphs

Application: Perfect Hash Function

Summary

CONCLUSION AND OPEN PROBLEMS

Shown:

- threshold for appearance of 2-core in mixed hypergraphs can be larger than 2-core threshold for k-uniform ones

Conclusion and Open Problems

Shown:

- threshold for appearance of 2-core in mixed hypergraphs can be larger than 2-core threshold for k-uniform ones
- threshold for two edge sizes can be determined efficiently

Conclusion and Open Problems

Shown:

- threshold for appearance of 2-core in mixed hypergraphs can be larger than 2-core threshold for k-uniform ones
- threshold for two edge sizes can be determined efficiently
- can be used to improve data structures based on random hash functions

CONCLUSION AND OPEN PROBLEMS

Shown:

- threshold for appearance of 2-core in mixed hypergraphs can be larger than 2-core threshold for k-uniform ones
- threshold for two edge sizes can be determined efficiently
- can be used to improve data structures based on random hash functions

Open:

- Given an upper bound $\overline{K}$ for the average edge size. Which pair $\boldsymbol{k}$ and $\boldsymbol{\alpha}$ maximizes the threshold $c^*(\boldsymbol{k}, \boldsymbol{\alpha})$ under the condition that $\boldsymbol{\alpha} \cdot \boldsymbol{k} \leq \overline{K}$?

CONCLUSION AND OPEN PROBLEMS

Shown:

- threshold for appearance of 2-core in mixed hypergraphs can be larger than 2-core threshold for k-uniform ones
- threshold for two edge sizes can be determined efficiently
- can be used to improve data structures based on random hash functions

Open:

- Given an upper bound $\overline{K}$ for the average edge size. Which pair $\boldsymbol{k}$ and $\boldsymbol{\alpha}$ maximizes the threshold $c^*(\boldsymbol{k}, \boldsymbol{\alpha})$ under the condition that $\boldsymbol{\alpha} \cdot \boldsymbol{k} \leq \overline{K}$?
- Prove or disprove:

  Conjecture ([Panagiotou, 2012])

  For each $\varepsilon > 0$ there exists $\boldsymbol{k}, \boldsymbol{\alpha}$ such that $c^*(\boldsymbol{k}, \boldsymbol{\alpha}) \geq 1 - \varepsilon$.

Thank you!

Definition of Key Function T(z)

The original "key function" is defined as

$$t(\lambda; \boldsymbol{k}, \boldsymbol{\alpha}) = \frac{\lambda}{\sum_{i=1}^{s} \alpha_i \cdot k_i \cdot \left(\Pr\left(\mathrm{Po}\left[\lambda\right] \geq 1 \right) \right)^{k_i - 1}} ,$$

for $\lambda \in (0, +\infty)$.

We transform $t(\lambda; \boldsymbol{k}, \boldsymbol{\alpha})$ in a more manageable function using a monotonic and bijective domain mapping via $z = 1 - e^{-\lambda}$ and $\lambda = -\ln(1 - z)$.

Hence the transformed "key function" is

$$T(z; \boldsymbol{k}, \boldsymbol{\alpha}) = \frac{-\ln(1 - z)}{\sum_{i=1}^{s} \alpha_i \cdot k_i \cdot z^{k_i - 1}} ,$$

for $z \in (0, 1)$.

Phase 1

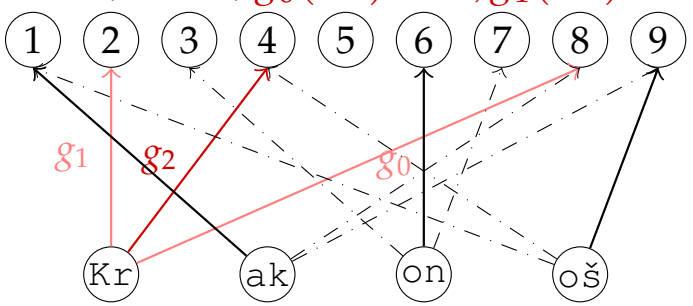
Left-perfect matching:

- given m keys $x_1, x_2, \ldots, x_m$ and number r

  $m = 4, r = 9, x_1 = \texttt{Kr}, x_2 = \texttt{ak}, x_3 = \texttt{on}, x_4 = \texttt{oš}$

- build bipartite graph $\mathcal{B}$
  - node sets $\{x_1, x_2, \ldots, x_m\}$ and $\{1, 2, \ldots, r\}$
  - edges given via d random hash functions $g_0, g_1, \ldots, g_{d-1}$

    $d = 3, r = 9, g_0(\texttt{Kr}) = 2, g_1(\texttt{Kr}) = 4, g_2(\texttt{Kr}) = 8$



- determine matching in $\mathcal{B}$, for each matching edge $\{x_i, g_{v_i}(x_i)\}$ temporarily store key-index pair (x_i, v_i)

  $(\texttt{Kr}, 2), (\texttt{ak}, 0), (\texttt{on}, 1), (\texttt{oš}, 0)$

PHASE 2

Retrieval:

- given key-index pairs (x_i, v_i), $i = 1, 2, \ldots, m$, and group $(\mathbb{Z}_d, +)$

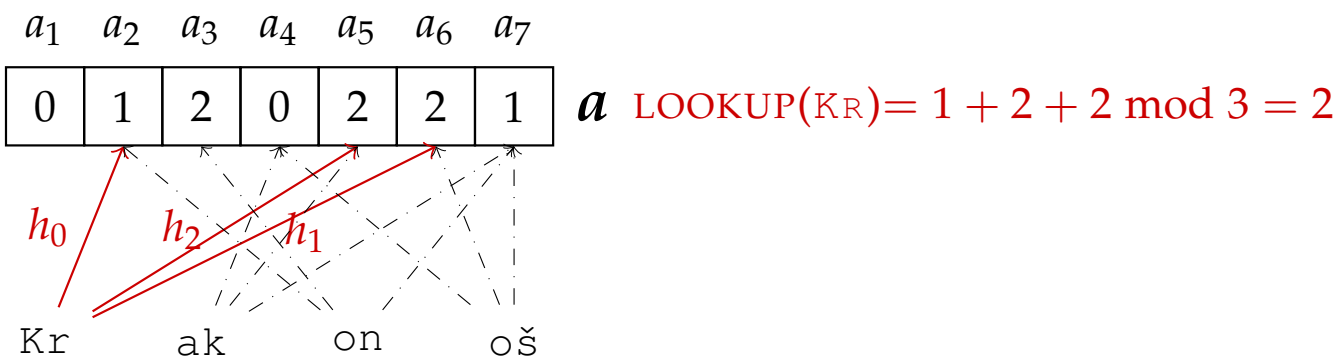
  $(\texttt{Kr}, 2), (\texttt{ak}, 0), (\texttt{on}, 1), (\texttt{oš}, 0), (\mathbb{Z}_3, +)$

- build binary $m \times n$ matrix $\boldsymbol{M}$, where the position of ones in row i are given via the values of random hash functions $h_0, h_1, \ldots, h_{k-1}$ for key x_i

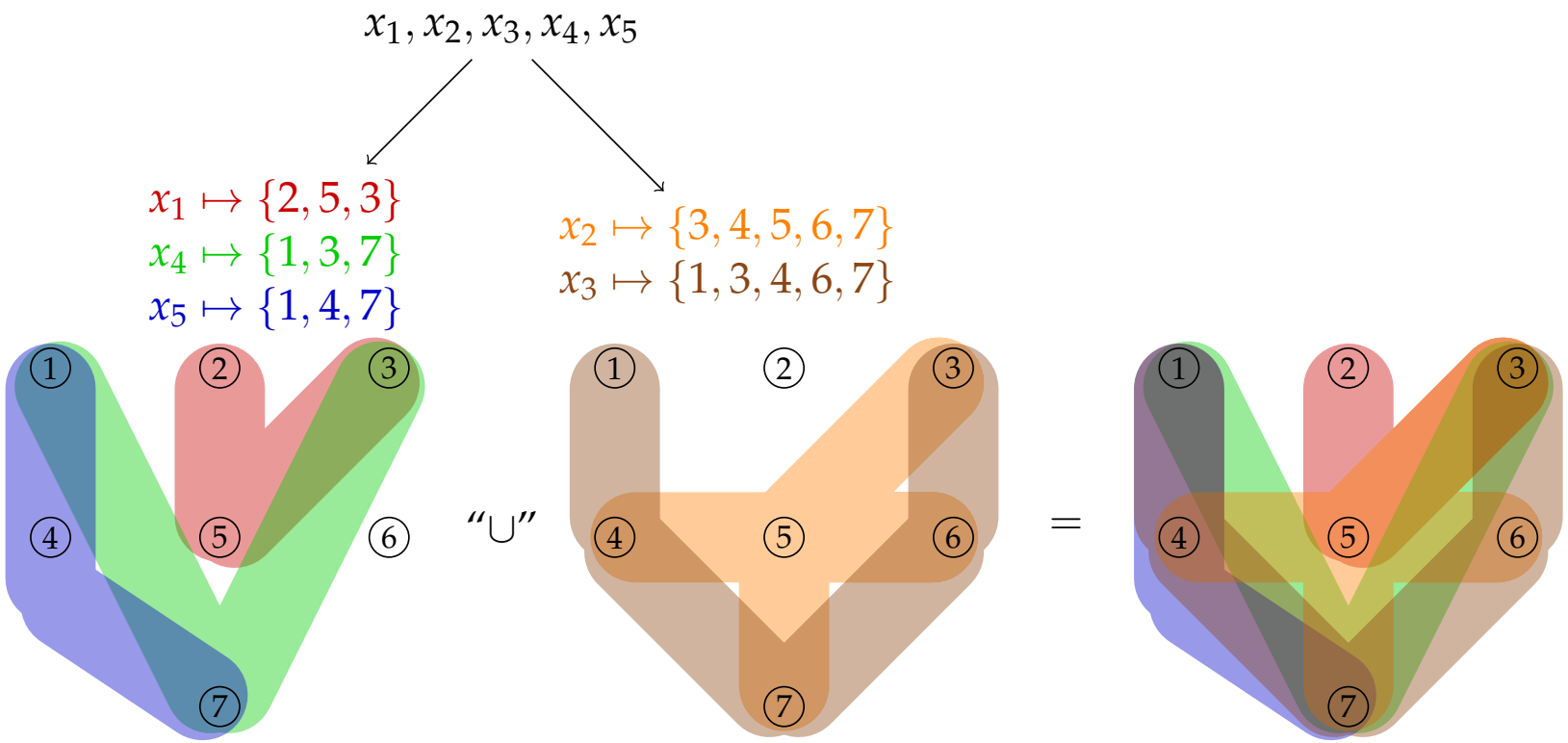
  $k = 3, n = 7, h_0(\texttt{Kr}) = 2, h_1(\texttt{Kr}) = 6, h_2(\texttt{Kr}) = 5 \Rightarrow \texttt{Kr} \mapsto (0\,1\,0\,0\,1\,1\,0)$

- solve $\boldsymbol{M} \cdot \boldsymbol{a} = \boldsymbol{v}$

- $f := g_v(x)$, where
  $v = \text{LOOKUP}(x) := a_{h_0(x)} \oplus a_{h_1(x)} \oplus \cdots \oplus a_{h_{k-1}(x)}$



Mixed Hypergraphs



References (1)


Botelho, F. C., Pagh, R., and Ziviani, N. (2007).
Simple and Space-Efficient Minimal Perfect Hash Functions.
In *Proc. 10th WADS*, volume 4619 of *LNCS*, pages 139–150. Springer.


Chazelle, B., Kilian, J., Rubinfeld, R., and Tal, A. (2004).
The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables.
In *Proc. 15th SODA*, pages 30–39. SIAM.


Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., and Rink, M. (2010).
Tight Thresholds for Cuckoo Hashing via XORSAT.
In *Proc. 37th ICALP (1)*, volume 6198 of *LNCS*, pages 213–225. Springer.

References (2)


Dietzfelbinger, M. and Rink, M. (2012).
Towards Optimal Degree-Distributions for Left-Perfect Matchings in Random Bipartite Graphs.
In *Proc. 7th CSR*, volume 7353 of *LNCS*, pages 99–111. Springer.


Fountoulakis, N. and Panagiotou, K. (2010).
Orientability of Random Hypergraphs and the Power of Multiple Choices.
In *Proc. 37th ICALP (1)*, volume 6198 of *LNCS*, pages 348–359. Springer.


Frieze, A. M. and Melsted, P. (2009).
Maximum Matchings in Random Bipartite Graphs and the Space Utilization of Cuckoo Hashtables.
CoRR, abs/0910.5535.
http://arxiv.org/abs/0910.5535.

References (3)


Goodrich, M. T. and Mitzenmacher, M. (2011).
Invertible Bloom Lookup Tables.
In *Proc. 49th Communication, Control, and Computing (Allerton)*, pages 792–799.


Luby, M. (2002).
LT Codes.
In *Proc. 43rd FOCS*, pages 271–.


Luby, M., Mitzenmacher, M., Shokrollahi, M. A., and Spielman, D. A. (2001).
Efficient Erasure Correcting Codes.
IEEE Transactions on Information Theory, 47(2):569–584.


Majewski, B. S., Wormald, N. C., Havas, G., and Czech, Z. J. (1996).
A Family of Perfect Hashing Methods.
Comput. J., 39(6):547–554.

References (4)


Maymounkov, P. (2002).
Online codes (Extended Abstract).
Technical Report TR2002-833, New York University.


Mitzenmacher, M. and Varghese, G. (2012).
Biff (Bloom Filter) Codes: Fast Error Correction for Large Data Sets.
In *Proc. ISIT 2012*, pages 483–487.


Molloy, M. (2004).
The pure literal rule threshold and cores in random hypergraphs.
In *Proc. 15th SODA*, pages 672–681. SIAM.


Panagiotou, K. (2012).
personal communication.

References (5)


Shokrollahi, A. (2006).
Raptor Codes.
IEEE Transactions on Information Theory, 52(6):2551–2567.