Matchings in Random Bipartite Graphs with Applications to Hashing-based Data Structures

Michael Rink

TECHNISCHE UNIVERSITÄT
ILMENAU

Motivation

Given: set of n keys $S := \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe)

Motivation

Given: set of n keys $S \coloneqq \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe)

Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
`lookup`$(\mathfrak{D}, x)$ returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$

Motivation

Given: set of n keys $S := \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe)

Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
`lookup(`$\mathfrak{D}, x$`)` returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$
1	0	membership tester

Motivation

Given: set of n keys $S \coloneqq \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe)

Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
$\texttt{lookup}(\mathfrak{D}, x)$ returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$
1	0	membership tester
$i_x \in [m] \coloneqq \{0, 1, \ldots, m-1\}$ $\forall x \in S$ pairwise distinct	arbitrary	injective mapping

Motivation

Given: set of n keys $S \coloneqq \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe), or
n key-value pairs $f \coloneqq \{(x_0, v_0), (x_1, v_1), \ldots, (x_{n-1}, v_{n-1})\}$

Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
`lookup`$(\mathfrak{D}, x)$ returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$
1	0	membership tester
$i_x \in [m] \coloneqq \{0, 1, \ldots, m-1\}$ $\forall x \in S$ pairwise distinct	arbitrary	injective mapping

Motivation

Given: set of n keys $S \coloneqq \{x_0, x_1, \dots, x_{n-1}\}$, $S \subseteq U$ (finite universe), or
n key-value pairs $f \coloneqq \{(x_0, v_0), (x_1, v_1), \dots, (x_{n-1}, v_{n-1})\}$

Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
`lookup(`$\mathfrak{D}, x$`)` returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$
1	0	membership tester
$i_x \in [m] \coloneqq \{0, 1, \dots, m-1\}$ $\forall x \in S$ pairwise distinct	arbitrary	injective mapping
$f(x)$	"$x \notin S$"	dictionary

Motivation

Given: set of n keys $S \coloneqq \{x_0, x_1, \ldots, x_{n-1}\}$, $S \subseteq U$ (finite universe), or
n key-value pairs $f \coloneqq \{(x_0, v_0), (x_1, v_1), \ldots, (x_{n-1}, v_{n-1})\}$

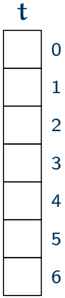
Task: build a static data structure $\mathfrak{D}$, where for each $x \in U$
`lookup`$(\mathfrak{D}, x)$ returns

$x \in S$	$x \in U \setminus S$	$\mathfrak{D}$
1	0	membership tester
$i_x \in [m] \coloneqq \{0, 1, \ldots, m-1\}$ $\forall x \in S$ pairwise distinct	arbitrary	injective mapping
$f(x)$	"$x \notin S$"	dictionary
$f(x)$	arbitrary	retrieval DS

Basic Scheme

Structure:

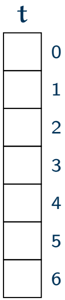
- table t with m cells, each of capacity r bits



Basic Scheme

Structure:

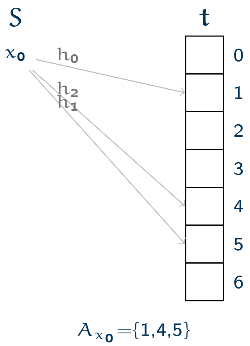
- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$



Basic Scheme

Structure:

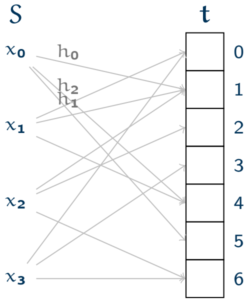
- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)



Basic Scheme

Structure:

- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)



$A_{x_0}=\{1,4,5\}$

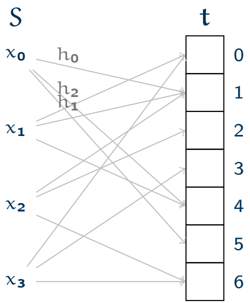
$n=4, m=7, d=3$

Basic Scheme

Structure:

- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)

Assumption: hash functions are ideal



$A_{x_0}=\{1,4,5\}$

$n=4, m=7, d=3$

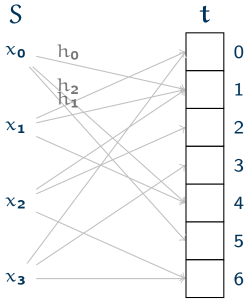
Basic Scheme

Structure:

- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)

Assumption: hash functions are ideal

- fully random on S (uniform, independent)



$A_{x_0}=\{1,4,5\}$

$n=4, m=7, d=3$

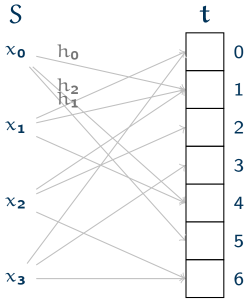
Basic Scheme

Structure:

- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)

Assumption: hash functions are ideal

- fully random on S (uniform, independent)
- negligible space needs



$A_{x_0}=\{1,4,5\}$

$n=4, m=7, d=3$

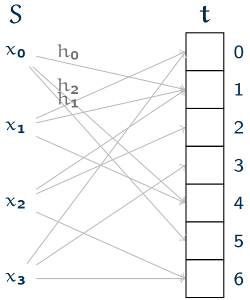
Basic Scheme

Structure:

- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)

Assumption: hash functions are ideal

- fully random on S (uniform, independent)
- negligible space needs
- constant evaluation time



$A_{x_0}=\{1,4,5\}$

$n=4, m=7, d=3$

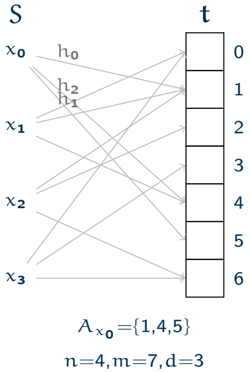
Basic Scheme

Structure:

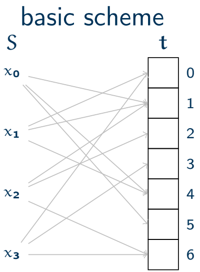
- table t with m cells, each of capacity r bits
- d hash functions $h_0, h_1, \ldots, h_{d-1} \colon U \to [m]$
- each x from U is mapped to set A_x of d addresses via the h_i's (*pairwise distinct* or with duplicates)

Assumption: hash functions are ideal

- fully random on S (uniform, independent)
- negligible space needs
- constant evaluation time
- e.g. [Dietzfelbinger and Rink, 2009] **not a topic here**

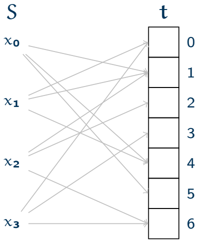


Abstract Representations

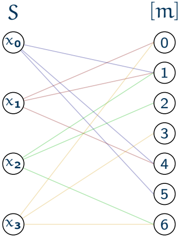


Abstract Representations

basic scheme

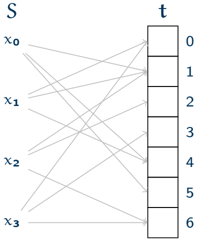


bipartite graph $\mathcal{B}$

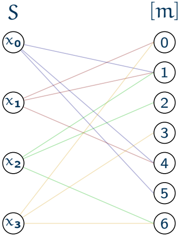


Abstract Representations

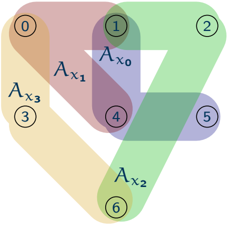
basic scheme



bipartite graph $\mathcal{B}$

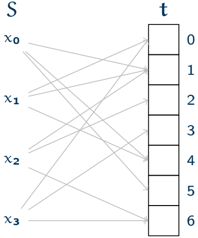


hypergraph $\mathcal{H}$

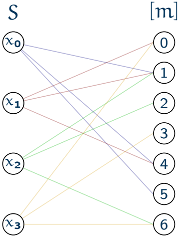


Abstract Representations

basic scheme



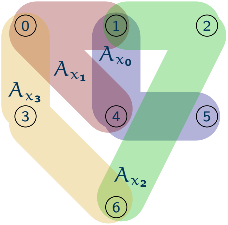
bipartite graph $\mathcal{B}$



binary matrix $\boldsymbol{M}$

$$
\begin{array}{c}
\\
A_{x_0} \\
A_{x_1} \\
A_{x_2} \\
A_{x_3}
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\end{array}
$$

hypergraph $\mathcal{H}$



Topic

Interested in:

- maximum $c = n/m$, $c = c(d)$, such that construction is successful with high probability (fixed d)
- (expected) construction time for t as function of n (fixed m and d)

Topic

Interested in:

- maximum $c = n/m$, $c = c(d)$, such that construction is successful with high probability (fixed d)
- (expected) construction time for t as function of n (fixed m and d)

Main Contributions:

- analysis for non-uniform left degrees (edge sizes, row weights)
- algorithm design

Topic

Interested in:

- maximum $c = n/m$, $c = c(d)$, such that construction is successful with high probability (fixed d)
- (expected) construction time for t as function of n (fixed m and d)

Main Contributions:

- analysis for non-uniform left degrees (edge sizes, row weights)
- algorithm design

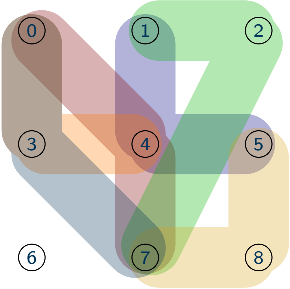
Measurements:

- time for $\texttt{lookup}(\mathfrak{D}, x)$ in number of cell probes
- space complexity in bits
- time complexity in word operations

Outline

Next ...

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

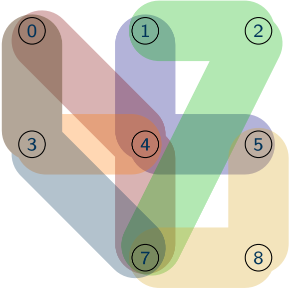
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

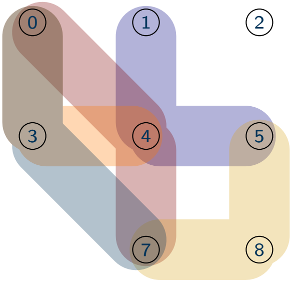
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

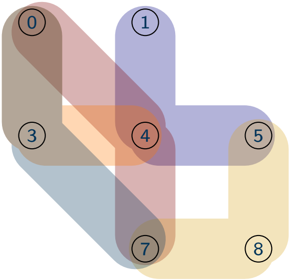
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

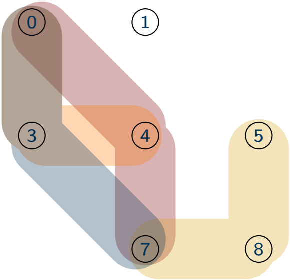
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

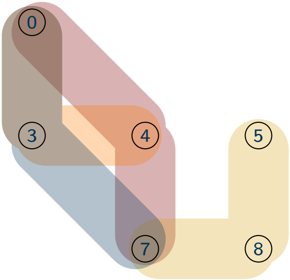
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

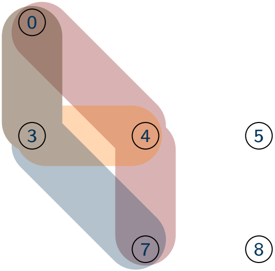
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

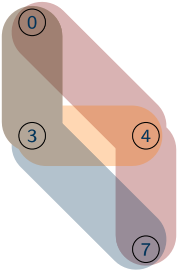
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

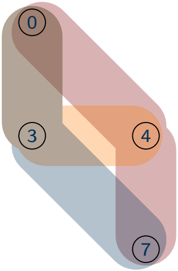
while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

The 2-Core



Algorithm: Peeling

Input: hypergraph $\mathcal{H} = ([m], E)$

Output: maximum induced sub-hypergraph with minimum degree 2

while $\mathcal{H}$ *has a node* v *of degree* $\leqslant 1$ **do**

  if v *is incident to an edge* A_x **then** delete A_x

  delete v

return $\mathcal{H}$

Analogous procedure in other formulations $\mathcal{B}$ and $\mathbf{M}$ gives the (equivalent) "2-core".

2-core Appearance and Density

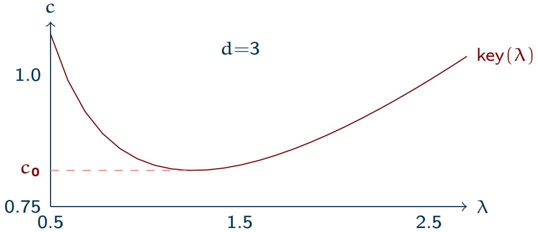
Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$.

2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" $\mathsf{key}(\lambda)$ with global minimum c_0 such that with high probability the following holds:

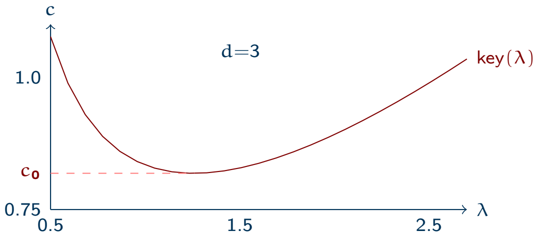


2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" key(λ) with global minimum c_0 such that with high probability the following holds:

- if $c < c_0$ then $\mathcal{H}$ has an empty 2-core,
- if $c > c_0$ then $\mathcal{H}$ has a non-empty 2-core.



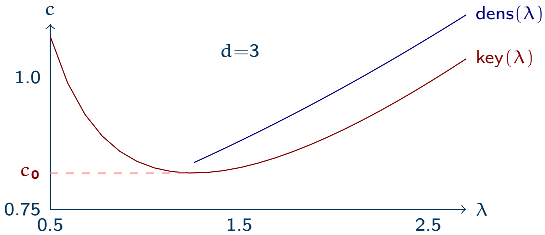
2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" key(λ) with global minimum c_0 such that with high probability the following holds:

- if $c < c_0$ then $\mathcal{H}$ has an empty 2-core,
- if $c > c_0$ then $\mathcal{H}$ has a non-empty 2-core.

There is a "density function" dens(λ) such that if $c > c_0$



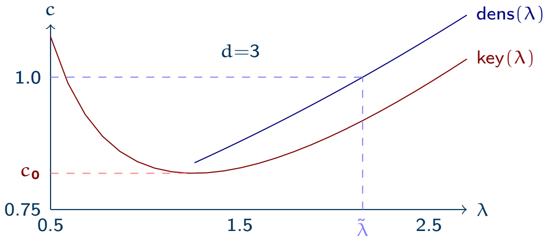
2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" $\text{key}(\lambda)$ with global minimum c_0 such that with high probability the following holds:

- if $c < c_0$ then $\mathcal{H}$ has an empty 2-core,
- if $c > c_0$ then $\mathcal{H}$ has a non-empty 2-core.

There is a "density function" $\text{dens}(\lambda)$ such that if $c > c_0$, then the edge density of the 2-core is tightly concentrated around $\text{dens}(\tilde{\lambda})$



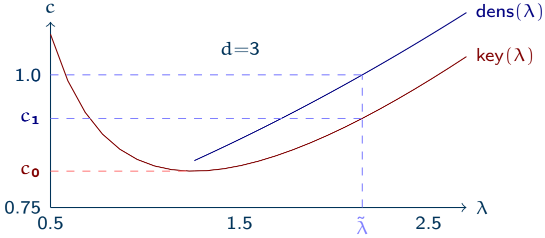
2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" key(λ) with global minimum c_0 such that with high probability the following holds:

- if $c < c_0$ then $\mathcal{H}$ has an empty 2-core,
- if $c > c_0$ then $\mathcal{H}$ has a non-empty 2-core.

There is a "density function" dens(λ) such that if $c > c_0$, then the edge density of the 2-core is tightly concentrated around dens$(\tilde{\lambda})$, where $\tilde{\lambda}$ is the solution of $c \overset{!}{=} \text{key}(\lambda)$.



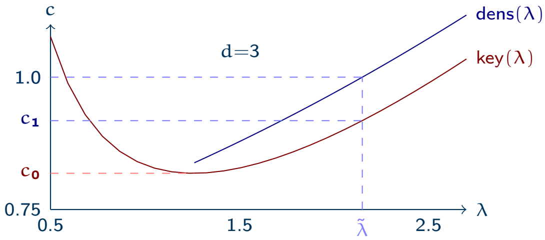
2-core Appearance and Density

Theorem: [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...

Let $d \geqslant 3$ and $n/m = c$. There is a convex "key function" key(λ) with global minimum c_0 such that with high probability the following holds:

- if $c < c_0$ then $\mathcal{H}$ has an empty 2-core,
- if $c > c_0$ then $\mathcal{H}$ has a non-empty 2-core.

There is a "density function" dens(λ) such that if $c > c_0$, then the edge density of the 2-core is tightly concentrated around dens$(\tilde{\lambda})$, where $\tilde{\lambda}$ is the solution of $c \overset{!}{=} \text{key}(\lambda)$.



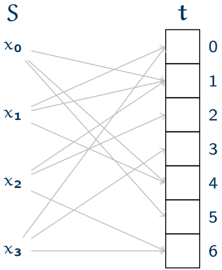
Interested in:
c_0 and c_1, the edge density of $\mathcal{H}$ where the edge density of its 2-core is 1

Next . . .

Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

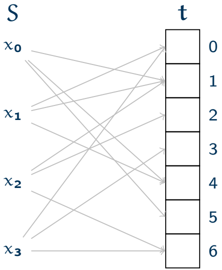
- membership tester:



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

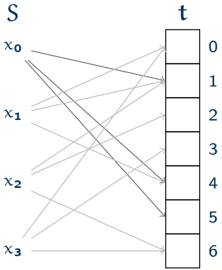
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

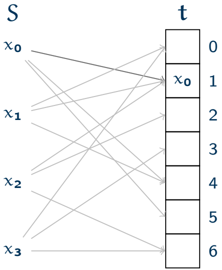
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

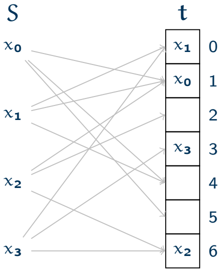
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

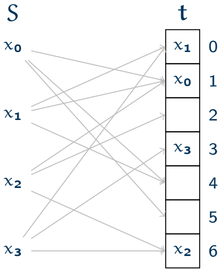
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

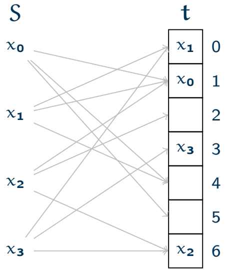
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x
  - $\texttt{lookup}(\mathfrak{D}, x) \coloneqq [\exists a \in A_x : t_a = x]$



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

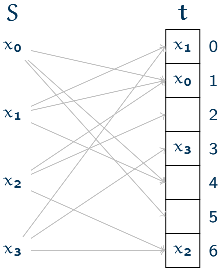
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x
  - $\texttt{lookup}(\mathfrak{D}, x) \coloneqq [\exists a \in A_x : t_a = x]$
- dictionary:
  - analogously, store tuple $(x, f(x))$



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

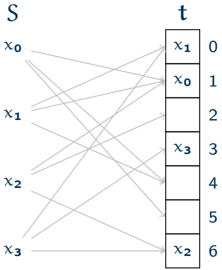
- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x
  - $\texttt{lookup}(\mathfrak{D}, x) := [\exists a \in A_x \colon t_a = x]$
- dictionary:
  - analogously, store tuple $(x, f(x))$
- maximum one entry per cell



Multiple Choice Hash Table

d-ary Cuckoo Hashing [Fotakis, Pagh, Sanders, Spirakis, 2003]**:**

- membership tester:
  - for all x from S store x in one cell of $\mathbf{t}$ whose address is from A_x
  - $\texttt{lookup}(\mathfrak{D}, x) \coloneqq [\exists a \in A_x : t_a = x]$
- dictionary:
  - analogously, store tuple $(x, f(x))$
- maximum one entry per cell
- cell probes: d worst-case, $(d+1)/2$ expected

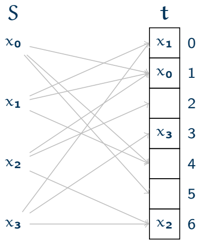


Requirements

Construction possible:

$\stackrel{\text{def}}{=}$ injective mapping $\sigma\colon S \to [m]$,
s.t. $\sigma(x) \in A_x$, for all $x \in S$

basic scheme



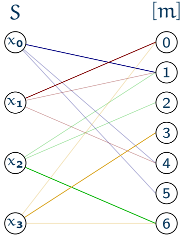
Requirements

Construction possible:

$\stackrel{\text{def}}{=}$ injective mapping $\sigma\colon S \to [m]$,
s.t. $\sigma(x) \in A_x$, for all $x \in S$

$\Leftrightarrow$ left-perfect matching in $\mathcal{B}$

bipartite graph $\mathcal{B}$



Requirements

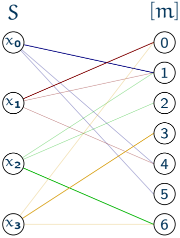
Construction possible:

$\overset{\text{def}}{=}$ injective mapping $\sigma\colon S \to [m]$, s.t. $\sigma(x) \in A_x$, for all $x \in S$

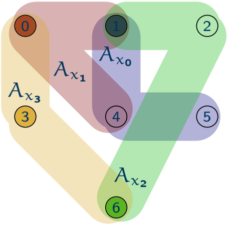
$\Leftrightarrow$ left-perfect matching in $\mathcal{B}$

$\Leftrightarrow$ edge orientation in $\mathcal{H}$ with indegree $\leqslant 1$

bipartite graph $\mathcal{B}$



hypergraph $\mathcal{H}$



Requirements

Construction possible:

$\overset{\text{def}}{=}$ injective mapping $\sigma\colon S \to [m]$, s.t. $\sigma(x) \in A_x$, for all $x \in S$

$\Leftrightarrow$ left-perfect matching in $\mathcal{B}$

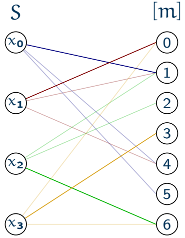
$\Leftrightarrow$ edge orientation in $\mathcal{H}$ with indegree $\leqslant 1$

$\Leftrightarrow$ $n \times n$ submatrix of $\boldsymbol{M} \geqslant$ permutation matrix

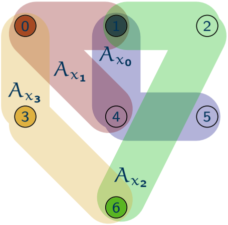
binary matrix $\boldsymbol{M}$

$$\begin{array}{c|ccccccc|}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
A_{x_0} & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
A_{x_1} & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
A_{x_2} & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
A_{x_3} & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
\hline
\end{array}$$

bipartite graph $\mathcal{B}$



hypergraph $\mathcal{H}$



Thresholds

Theorem: [Fotakis, Pagh, Sanders, Spirakis, 2003]

Let $c = n/m$. If $d = \Theta\left(\ln\left(\frac{1}{1-c}\right)\right)$, then with high probability (whp) $\mathcal{B}$ admits a left-perfect matching.

Thresholds

Theorem: [Fotakis, Pagh, Sanders, Spirakis, 2003]

Let $c = n/m$. If $d = \Theta\left(\ln\left(\frac{1}{1-c}\right)\right)$, then with high probability (whp) $\mathcal{B}$ admits a left-perfect matching.

Theorem: [Bohman and Kim, 2006, $d = 4$],[Frieze and Melsted, 2009], [Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh, Rink, 2010], [Fountoulakis and Panagiotou, 2010],

Let $c = n/m$ and let $d \geqslant 3$. Then whp the following holds:

Thresholds

Theorem: [Fotakis, Pagh, Sanders, Spirakis, 2003]

Let $c = n/m$. If $d = \Theta\left(\ln\left(\frac{1}{1-c}\right)\right)$, then with high probability (whp) $\mathcal{B}$ admits a left-perfect matching.

Theorem: [Bohman and Kim, 2006, $d = 4$],[Frieze and Melsted, 2009], [Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh, Rink, 2010], [Fountoulakis and Panagiotou, 2010],

Let $c = n/m$ and let $d \geqslant 3$. Then whp the following holds:

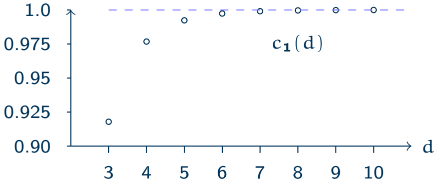
- if $c < c_1(d)$, then $\mathcal{B}$ admits a left-perfect matching.
- if $c > c_1(d)$, then $\mathcal{B}$ admits no left-perfect matching.

Thresholds

Theorem: [Fotakis, Pagh, Sanders, Spirakis, 2003]
Let $c = n/m$. If $d = \Theta\left(\ln\left(\frac{1}{1-c}\right)\right)$, then with high probability (whp) $\mathcal{B}$ admits a left-perfect matching.

Theorem: [Bohman and Kim, 2006, $d = 4$],[Frieze and Melsted, 2009], [Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh, Rink, 2010], [Fountoulakis and Panagiotou, 2010],
Let $c = n/m$ and let $d \geqslant 3$. Then whp the following holds:

- if $c < c_1(d)$, then $\mathcal{B}$ admits a left-perfect matching.
- if $c > c_1(d)$, then $\mathcal{B}$ admits no left-perfect matching.



New: Optimality

Question: Given a target average left degree $\overline{d}$, can we improve the success probability using different left degrees d_x compared to using the same left degree?

New: Optimality

Question: Given a target average left degree $\overline{d}$, can we improve the success probability using different left degrees d_x compared to using the same left degree?

$$\overline{d} = \frac{1}{n} \cdot \sum_{x \in S} d_x$$

New: Optimality

Question: Given a target average left degree $\overline{d}$, can we improve the success probability using different left degrees d_x compared to using the same left degree?

$$\overline{d} = \frac{1}{n} \cdot \sum_{x \in S} d_x$$

Theorem: [Dietzfelbinger and Rink, 2012, duplicates allowed]

Let $\overline{d} > 2$ be the average left degree of $\mathcal{B}$.

- If $\overline{d}$ is integral, then the optimal choice is $d_x = \overline{d}$ for all $x \in S$.
- If $\overline{d}$ is non-integral, then it is optimal if the fraction of nodes with degree $\begin{cases} \lfloor \overline{d} \rfloor \\ \lceil \overline{d} \rceil \end{cases}$ is tightly concentrated around $\begin{cases} \lceil \overline{d} \rceil - \overline{d} \\ \overline{d} - \lfloor \overline{d} \rfloor \end{cases}$.

New: Optimality

Question: Given a target average left degree $\overline{d}$, can we improve the success probability using different left degrees d_x compared to using the same left degree?

$$\overline{d} = \frac{1}{n} \cdot \sum_{x \in S} d_x$$

Theorem: [Dietzfelbinger and Rink, 2012, duplicates allowed]

Let $\overline{d} > 2$ be the average left degree of $\mathcal{B}$.

- If $\overline{d}$ is integral, then the optimal choice is $d_x = \overline{d}$ for all $x \in S$.
- If $\overline{d}$ is non-integral, then it is optimal if the fraction of nodes with degree $\begin{cases} \lfloor \overline{d} \rfloor \\ \lceil \overline{d} \rceil \end{cases}$ is tightly concentrated around $\begin{cases} \lceil \overline{d} \rceil - \overline{d} \\ \overline{d} - \lfloor \overline{d} \rfloor \end{cases}$.

Proof idea:

* Degree of each left node is random variable D_x with separate pmf.
* Fix $\mathcal{B}$, but omit 2 left nodes x, y. Compare probability for a matching under slight changes of pmf_x and pmf_y.

Thresholds for Mixed Degrees

Theorem: [Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh, Rink, 2010]

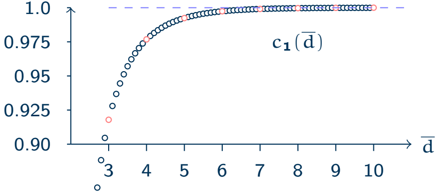
The results for uniform left degrees $d \geqslant 3$ can be extended to prove thresholds $c_1(\overline{d})$ for mixed left degrees $\overline{d} > 2$.

Thresholds for Mixed Degrees

Theorem: [Dietzfelbinger, Goerdt, Mitzenmacher, Montanari, Pagh, Rink, 2010]

The results for uniform left degrees $d \geqslant 3$ can be extended to prove thresholds $c_1(\overline{d})$ for mixed left degrees $\overline{d} > 2$.

left degrees d_x concentrated around $\overline{d}$



Standard Augmenting Path Algorithm

Hopcroft-Karp Algorithm [Hopcroft and Karp, 1973]**:**

Standard Augmenting Path Algorithm

Hopcroft-Karp Algorithm [Hopcroft and Karp, 1973]**:**

- Results by [Bast, Mehlhorn, Schäfer, Tamaki, 2004] indicate that in our situation the running time is $O(n \cdot \log n)$ whp.

Standard Augmenting Path Algorithm

Hopcroft-Karp Algorithm [Hopcroft and Karp, 1973]**:**

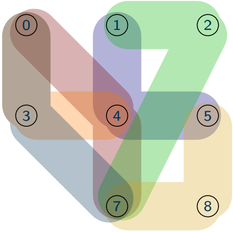
- Results by [Bast, Mehlhorn, Schäfer, Tamaki, 2004] indicate that in our situation the running time is $O(n \cdot \log n)$ whp.

Question: Can we do better, in linear time?

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



Algorithm: Generalized Selfless

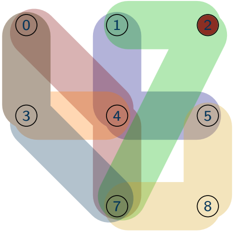
Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v

  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



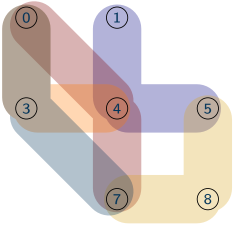
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



Algorithm: Generalized Selfless

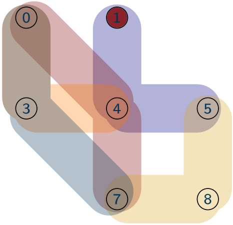
Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v

  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



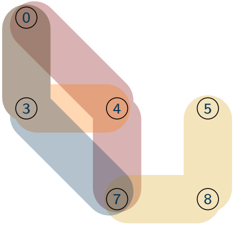
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]**:**

- adaption of "Selfless Algorithm" by [Sanders, 2004]



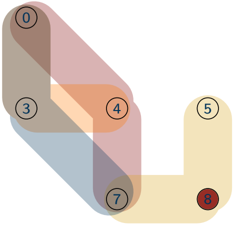
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



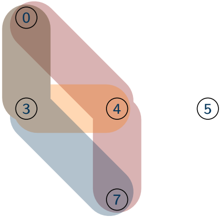
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



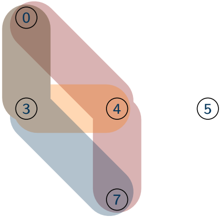
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$

Output: matching in $\mathcal{B}$

repeat

  if *a node* v *has degree* 1 **then** choose v

  else

    choose node v of minimum priority

    $$\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$$

  choose edge $A_x \ni v$

  match x and v

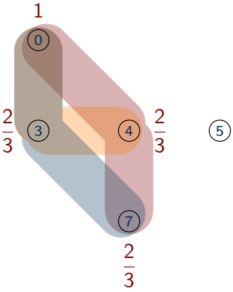
  delete A_x and v

until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



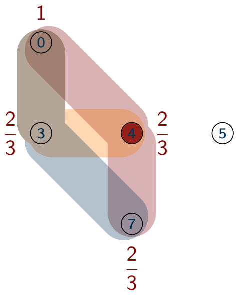
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  else
    choose node v of minimum priority
    $$\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$$
  choose edge $A_x \ni v$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$

Output: matching in $\mathcal{B}$

repeat

  if *a node* v *has degree* 1 **then** choose v

  else

    choose node v of minimum priority

$$\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$$

  choose edge $A_x \ni v$ with min cardinality $|A_x|$

  match x and v

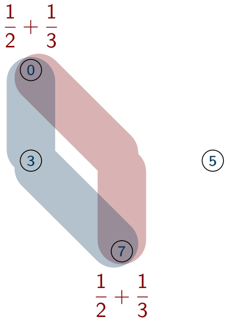
  delete A_x and v

until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



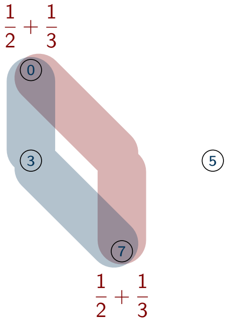
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  else
    choose node v of minimum priority
    $\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$
  choose edge $A_x \ni v$ with min cardinality $|A_x|$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

- adaption of "Selfless Algorithm" by [Sanders, 2004]



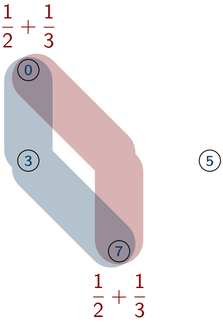
Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  else
    choose node v of minimum priority
    $\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$
    if *minimum priority* > 1 **then return** failure
  choose edge $A_x \ni v$ with min cardinality $|A_x|$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

Greedy Approach

Generalized Selfless Algorithm [Dietzfelbinger et al., 2010]:

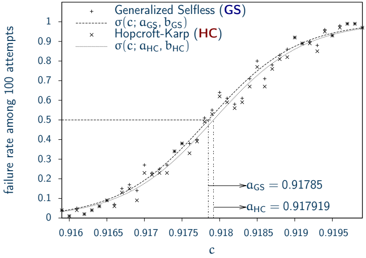
- adaption of "Selfless Algorithm" by [Sanders, 2004]
- running time $O(n)$



Algorithm: Generalized Selfless

Input: hypergraph $\mathcal{H}$
Output: matching in $\mathcal{B}$
repeat
  if *a node* v *has degree* 1 **then** choose v
  else
    choose node v of minimum priority
    $\pi(v) = \sum_{A_x \ni v} \frac{1}{|A_x|}$
    if *minimum priority* > 1 **then return** failure
  choose edge $A_x \ni v$ with min cardinality $|A_x|$
  match x and v
  delete A_x and v
until *all edges have been deleted at the end*

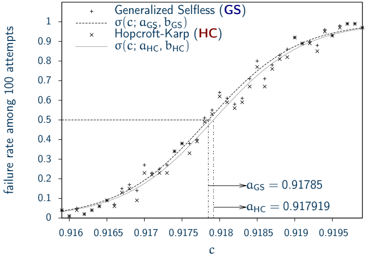
Heuristic vs Optimal Algorithm



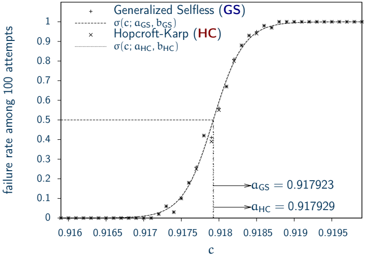
$m = 10^5$

$d = 3$; theoretical threshold $c_1(d) \approx 0.91794$, interval size 0.004

Heuristic vs Optimal Algorithm



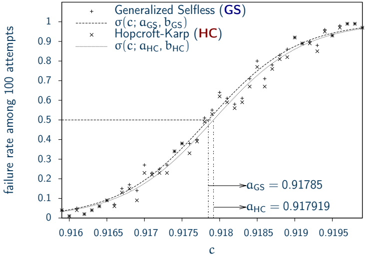
$m = 10^5$



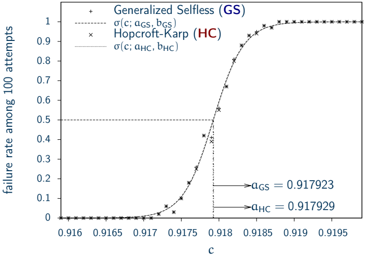
$m = 10^6$

$d = 3$; theoretical threshold $c_1(d) \approx 0.91794$, interval size 0.004

Heuristic vs Optimal Algorithm



$m = 10^5$



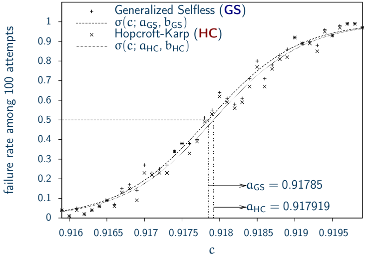
$m = 10^6$

$d = 3$; theoretical threshold $c_1(d) \approx 0.91794$, interval size 0.004

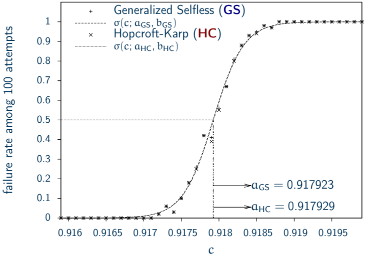
Running times for **GS** and **HC** in seconds on Intel Xeon 3GHz:

$m \backslash c$	0.916	0.917	0.918	0.919

Heuristic vs Optimal Algorithm



$m = 10^5$



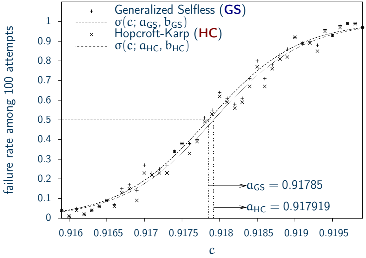
$m = 10^6$

$d = 3$; theoretical threshold $c_1(d) \approx 0.91794$, interval size 0.004

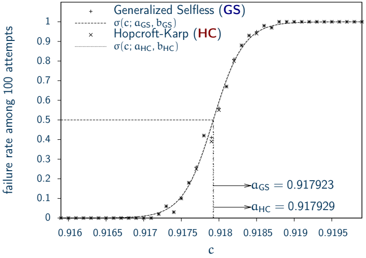
Running times for **GS** and **HC** in seconds on Intel Xeon 3GHz:

$m \backslash c$	0.916	0.917	0.918	0.919
10^5	**0.11** **0.64**	**0.11** **0.77**	**0.11** **0.88**	**0.11** **0.93**

Heuristic vs Optimal Algorithm



$m = 10^5$



$m = 10^6$

$d = 3$; theoretical threshold $c_1(d) \approx 0.91794$, interval size 0.004

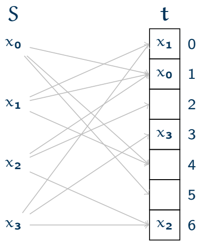
Running times for **GS** and **HC** in seconds on Intel Xeon 3GHz:

$m \backslash c$	0.916	0.917	0.918	0.919
10^5	**0.11** **0.64**	**0.11** **0.77**	**0.11** **0.88**	**0.11** **0.93**
10^6	**1.74** **9.34**	**1.75** **10.99**	**1.76** **16.36**	**1.76** **16.85**

Summary

Performance:

- space utilization
  - membership: $1/c \cdot n \cdot \log|U|$

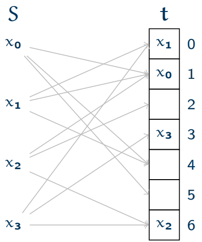


d	2	3	4	5	...	$O(\log(n))$
$1/c$	2.01	1.09	1.03	1.01	...	1

Summary

Performance:

- space utilization
  - membership: $1/c \cdot n \cdot \log|U|$
- construction time: $O(n)$
  (avg. in experiments)

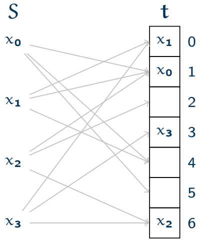


d	2	3	4	5	...	$O(\log(n))$
$1/c$	2.01	1.09	1.03	1.01	...	1

Summary

Performance:

- space utilization
  - membership: $1/c \cdot n \cdot \log|U|$
- construction time: $O(n)$ (avg. in experiments)
- cell probes: $O\left(\ln\left(\frac{1}{1-c}\right)\right)$

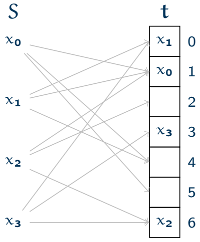


d	2	3	4	5	...	$O(\log(n))$
$1/c$	2.01	1.09	1.03	1.01	...	1

Summary

Performance:

- space utilization
  - membership: $1/c \cdot n \cdot \log|U|$
  - dictionary: $1/c \cdot n \cdot (\log|U| + \log|V|)$
- construction time: $O(n)$
  (avg. in experiments)
- cell probes: $O\left(\ln\left(\frac{1}{1-c}\right)\right)$

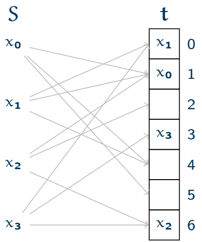


d	2	3	4	5	...	$O(\log(n))$
$1/c$	2.01	1.09	1.03	1.01	...	1

Summary

Performance:

- space utilization
  - membership: $1/c \cdot n \cdot \log|U|$
  - dictionary: $1/c \cdot n \cdot (\log|U| + \log|V|)$
- construction time: $O(n)$ (avg. in experiments)
- cell probes: $O\left(\ln\left(\frac{1}{1-c}\right)\right)$



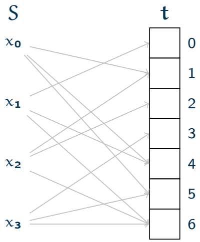
d	2	3	4	5	...	$O(\log(n))$
$1/c$	2.01	1.09	1.03	1.01	...	1

Open: Proof that if $\mathcal{B}$ admits a matching, then whp the Generalized Selfless Algorithm finds a matching.

Next . . .

Retrieval Data Structure

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Basic Retrieval Data Structure [Dietzfelbinger and Pagh, 2008]**:**

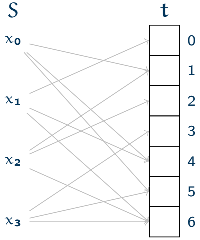


Retrieval Data Structure

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Basic Retrieval Data Structure [Dietzfelbinger and Pagh, 2008]**:**

- Assume: $(V, \oplus)$ is an abelian group

$(V, \oplus) = (\mathbb{Z}_6, +)$



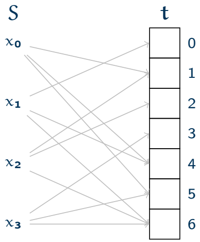
Retrieval Data Structure

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Basic Retrieval Data Structure [Dietzfelbinger and Pagh, 2008]**:**

- Assume: $(V, \oplus)$ is an abelian group
- Given $f\colon S \to V$, build vector $\boldsymbol{v} = (f(x_i))_{i \in [n]}$

$(V,\oplus)=(\mathbb{Z}_6,+)$

$\boldsymbol{v}=(2,1,5,5)$



Retrieval Data Structure

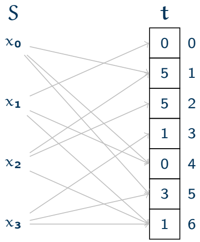
Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Basic Retrieval Data Structure [Dietzfelbinger and Pagh, 2008]**:**

- Assume: $(V, \oplus)$ is an abelian group
- Given $f\colon S \to V$, build vector $\boldsymbol{v} = (f(x_i))_{i\in[n]}$ and solve linear system

$(V,\oplus)=(\mathbb{Z}_6,+)$

$\boldsymbol{v}=(2,1,5,5)$

$$\mathbf{M} \cdot \mathbf{t} = \boldsymbol{v}$$



Retrieval Data Structure

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Basic Retrieval Data Structure [Dietzfelbinger and Pagh, 2008]**:**

- Assume: $(V, \oplus)$ is an abelian group
- Given $f\colon S \to V$, build vector $\boldsymbol{v} = (f(x_i))_{i\in[n]}$ and solve linear system

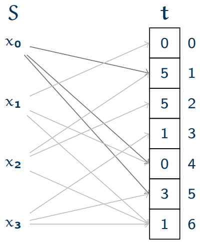
$$\mathbf{M} \cdot \mathbf{t} = \boldsymbol{v}$$

- $\mathtt{lookup}(\mathfrak{D}, x) \coloneqq \bigoplus_{a\in A_x} t_a$

$(V,\oplus)=(\mathbb{Z}_6,+)$

$\boldsymbol{v}=(2,1,5,5)$

$(5+0+3) \bmod 6=2$



Injective Mapping

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

Injective Mapping

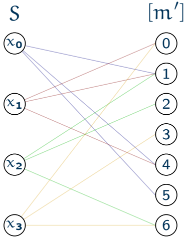
Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

- given S and range m' of injective mapping

Injective Mapping

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

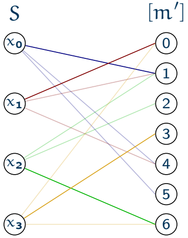
- given S and range m' of injective mapping
- build bipartite graph $\mathcal{B}'$
  - left node set S, right node set $[m']$
  - edges given via hash functions $h'_i(x)$, $i \in [d']$



Injective Mapping

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

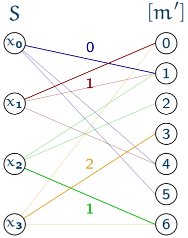
- given S and range m' of injective mapping
- build bipartite graph $\mathcal{B}'$
  - left node set S, right node set $[m']$
  - edges given via hash functions $h'_i(x)$, $i \in [d']$
- determine matching in $\mathcal{B}'$



Injective Mapping

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

- given S and range m' of injective mapping
- build bipartite graph $\mathcal{B}'$
  - left node set S, right node set $[m']$
  - edges given via hash functions $h'_i(x)$, $i \in [d']$
- determine matching in $\mathcal{B}'$
- build vector v of indices $\iota(x)$, where $\{x, h'_{\iota(x)}\}$ is matching edge

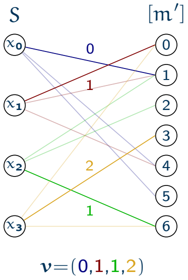


$v = (0, 1, 1, 2)$

Injective Mapping

Bloomier Filter [Chazelle, Kilian, Rubinfeld, Tal, 2004],
Perfect Hash Function [Botelho, Pagh, Ziviani, 2007], more general [Rink, 2013]**:**

- given S and range m' of injective mapping
- build bipartite graph $\mathcal{B}'$
  - left node set S, right node set $[m']$
  - edges given via hash functions $h'_i(x)$, $i \in [d']$
- determine matching in $\mathcal{B}'$
- build vector $\boldsymbol{v}$ of indices $\iota(x)$, where $\{x, h'_{\iota(x)}\}$ is matching edge
- build retrieval data structure for $\boldsymbol{v}$



Requirements (1)

Construction possible:

$\Leftarrow$ M has full row rank n, i.e. $n \times n$ submatrix with non-zero determinant in $\mathbb{F}_2$

Connections

Theorem: [Dietzfelbinger et al., 2010] based on [Dubois and Mandler, 2002]

The density threshold $c = n/m$ up to which whp $\boldsymbol{M}$ has full row rank is equivalent to c_1, the threshold where whp the edge density of the 2-core of $\mathcal{H}$ grows beyond 1.

Connections

Theorem: [Dietzfelbinger et al., 2010] based on [Dubois and Mandler, 2002]
The density threshold $c = n/m$ up to which whp $\boldsymbol{M}$ has full row rank is equivalent to c_1, the threshold where whp the edge density of the 2-core of $\mathcal{H}$ grows beyond 1.

Problem: Solving a linear system is harder than determining a matching.

- general upper bound $O(n^3)$ by Gaussian elimination
- in our situation maybe $O(n^2)$, e.g. [Wiedemann, 1986]

Connections

Theorem: [Dietzfelbinger et al., 2010] based on [Dubois and Mandler, 2002]

The density threshold $c = n/m$ up to which whp $\boldsymbol{M}$ has full row rank is equivalent to c_1, the threshold where whp the edge density of the 2-core of $\mathcal{H}$ grows beyond 1.

Problem: Solving a linear system is harder than determining a matching.

- general upper bound $O(n^3)$ by Gaussian elimination
- in our situation maybe $O(n^2)$, e.g. [Wiedemann, 1986]

Question: How can we reach linear running time?

Requirements (2)

Construction possible:

$\Leftarrow$ M has full row rank n

$$\begin{array}{c|ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline x_0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ x_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ x_2 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ x_3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

Requirements (2)

Construction possible:

- $\Leftarrow$ M has full row rank n
- $\Leftrightarrow$ elementary operations transform M in row echelon form

$$
\begin{array}{c c}
 & \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}
\end{array}
$$

Requirements (2)

Construction possible:

- $\Leftarrow$ $\boldsymbol{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\boldsymbol{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\boldsymbol{M}$ in row echelon form

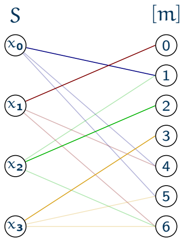
$$
\begin{array}{c c}
 & \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}
\end{array}
$$

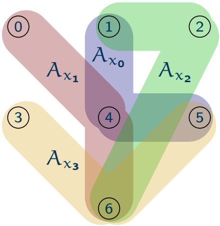
Requirements (2)

Construction possible:

- $\Leftarrow$ $\mathbf{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\mathbf{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\mathbf{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array}\begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$



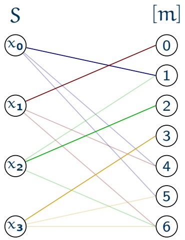


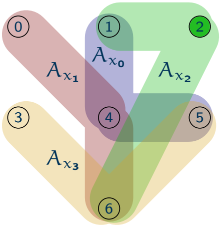
Requirements (2)

Construction possible:

- $\Leftarrow$ $\boldsymbol{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\boldsymbol{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\boldsymbol{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$\begin{array}{cc} & \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$



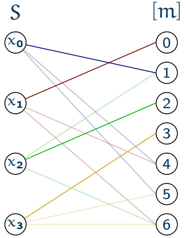


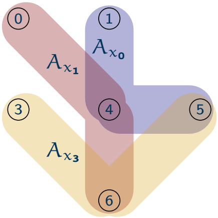
Requirements (2)

Construction possible:

- $\Leftarrow$ $\boldsymbol{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\boldsymbol{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\boldsymbol{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$
\begin{array}{c}
 \\ x_2 \\ x_1 \\ x_0 \\ x_3
\end{array}
\begin{array}{c}
\begin{matrix} 2 & 1 & 0 & 3 & 4 & 5 & 6 \end{matrix} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}
\end{array}
$$



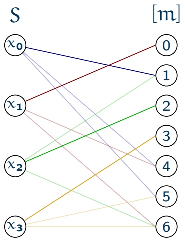


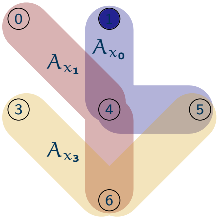
Requirements (2)

Construction possible:

- $\Leftarrow$ $\mathbf{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\mathbf{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\mathbf{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$\begin{array}{c} \\ x_2 \\ x_1 \\ x_0 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} 2 & 1 & 0 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$



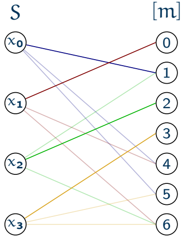


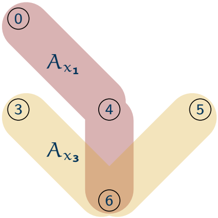
Requirements (2)

Construction possible:

- $\Leftarrow$ $\boldsymbol{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\boldsymbol{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\boldsymbol{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$\begin{array}{c|c} & \begin{array}{ccccccc} 2 & 1 & 0 & 3 & 4 & 5 & 6 \end{array} \\ \hline \begin{array}{c} x_2 \\ x_0 \\ x_1 \\ x_3 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$



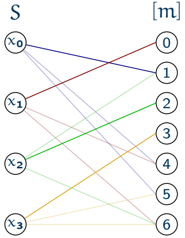


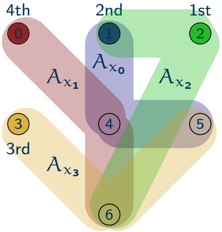
Requirements (2)

Construction possible:

- $\Leftarrow$ $\mathbf{M}$ has full row rank n
- $\Leftrightarrow$ elementary operations transform $\mathbf{M}$ in row echelon form
- $\Leftarrow$ *only* row and column permutations transform $\mathbf{M}$ in row echelon form
- $\Leftrightarrow$ 2-core of $\mathcal{H}$ is empty

$$\begin{array}{c|c} & \begin{array}{ccccccc} 2 & 1 & 3 & 0 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} x_2 \\ x_0 \\ x_3 \\ x_1 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$





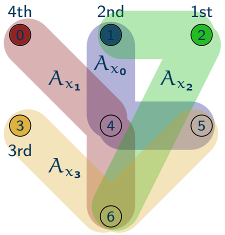
Peeling and Back-substitution

Greedy Algorithm:

Peeling and Back-substitution

Greedy Algorithm:

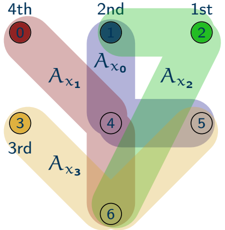
- determine row and column permutations



Peeling and Back-substitution

Greedy Algorithm:

- determine row and column permutations
- apply back-substitution

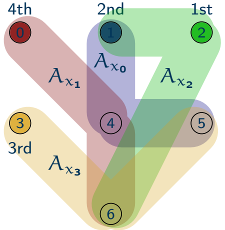


$$\begin{array}{c c}
 & \begin{array}{ccccccc} 2 & 1 & 3 & 0 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} x_2 \\ x_0 \\ x_3 \\ x_1 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}
\end{array}$$

Peeling and Back-substitution

Greedy Algorithm:

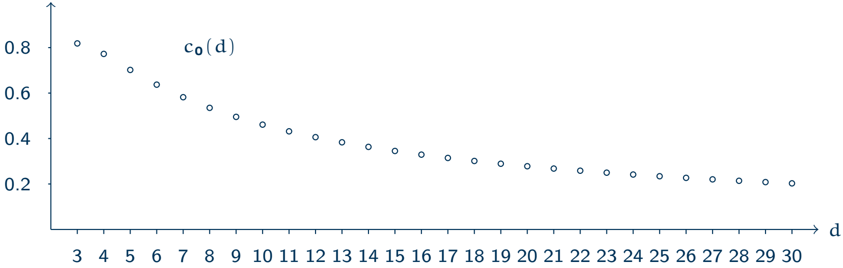
- determine row and column permutations
- apply back-substitution
- running time $O(n)$



$$\begin{array}{c c} & \begin{matrix} 2 & 1 & 3 & 0 & 4 & 5 & 6 \end{matrix} \\ \begin{matrix} x_2 \\ x_0 \\ x_3 \\ x_1 \end{matrix} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Thresholds

Appearance of 2-core [Molloy, 2004],[Cooper, 2004],[Kim, 2006],...



New: Optimality

Question: Can we beat $c_0(3) \approx 0.8185$ using different edge sizes?

New: Optimality

Question: Can we beat $c_0(3) \approx 0.8185$ using different edge sizes?

Theorem: [Rink, 2013] based on [Dietzfelbinger et al., 2010]

The analysis for the appearance of 2-cores in uniform hypergraphs can be extended to non-uniform hypergraphs with $\alpha_i \cdot n$ edges of size $d_i \geqslant 3$, leading to thresholds $c_0(\mathbf{d}, \boldsymbol{\alpha})$.

New: Optimality

Question: Can we beat $c_0(3) \approx 0.8185$ using different edge sizes?

Theorem: [Rink, 2013] based on [Dietzfelbinger et al., 2010]

The analysis for the appearance of 2-cores in uniform hypergraphs can be extended to non-uniform hypergraphs with $\alpha_i \cdot n$ edges of size $d_i \geqslant 3$, leading to thresholds $c_0(\mathbf{d}, \boldsymbol{\alpha})$.

Theorem: [Rink, 2013]

For two edge sizes d_0 and d_1 the maximum threshold

$$c_0(d_0, d_1) \coloneqq \max_{\alpha} c_0((d_0, d_1), (\alpha, 1 - \alpha))$$

can be calculated efficiently, and for appropriate d_0 and d_1, this value is larger than $c_0(3)$.

New: Optimality

Question: Can we beat $c_0(3) \approx 0.8185$ using different edge sizes?

Theorem: [Rink, 2013] based on [Dietzfelbinger et al., 2010]

The analysis for the appearance of 2-cores in uniform hypergraphs can be extended to non-uniform hypergraphs with $\alpha_i \cdot n$ edges of size $d_i \geqslant 3$, leading to thresholds $c_0(\mathbf{d}, \boldsymbol{\alpha})$.

Theorem: [Rink, 2013]

For two edge sizes d_0 and d_1 the maximum threshold

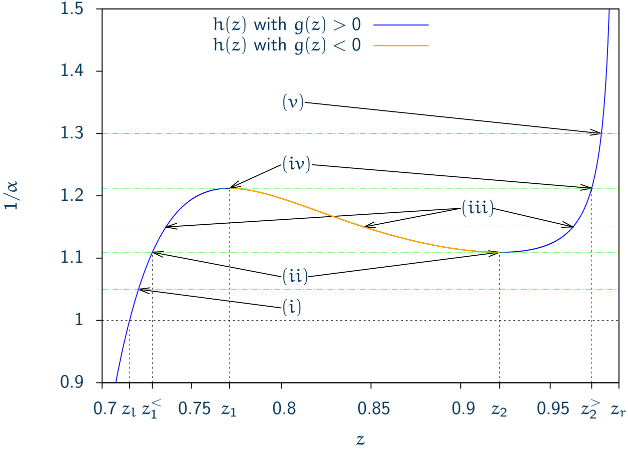
$$c_0(d_0, d_1) \coloneqq \max_{\alpha} c_0((d_0, d_1), (\alpha, 1-\alpha))$$

can be calculated efficiently, and for appropriate d_0 and d_1, this value is larger than $c_0(3)$.

Proof idea:

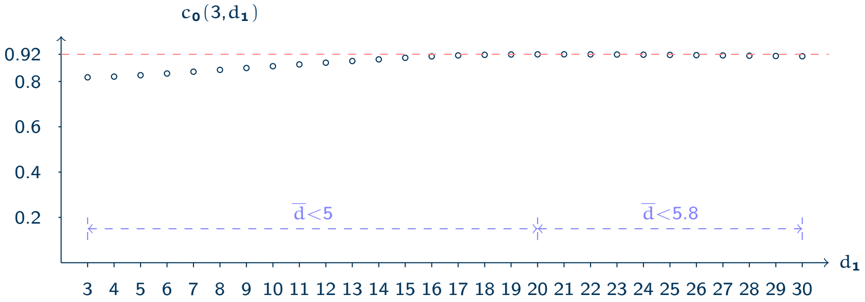
* multivariate calculus, non-convex optimization

Non-convex Optimization



- identify *critical points* in z-direction
- determine $z'(d_0, d_1)$, the maximum point of the *function of critical points*
- find z with smallest distance to z' that is *legal global minimum point*

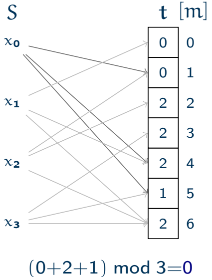
Thresholds for Mixed Degrees



Summary

Performance: e.g. using factor $1.1 > \dfrac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$

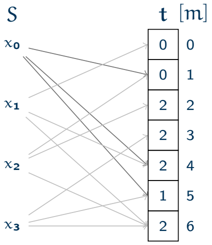


Summary

Performance: e.g. using factor $1.1 > \dfrac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$

- construction time: $O(n)$ (expected)



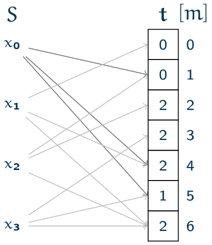
$(0+2+1) \bmod 3 = 0$

Summary

Performance: e.g. using factor $1.1 > \dfrac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$

- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16

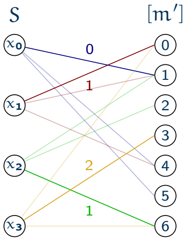


$(0+2+1) \bmod 3 = 0$

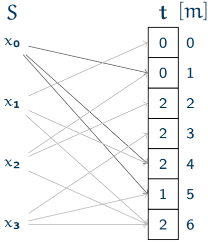
Summary

Performance: e.g. using factor $1.1 > \frac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$
  - injective mapping: $m' = m = 1.1 \cdot n \Rightarrow d' = 3$ hash functions for $\mathcal{B}'$
- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16



$v=(0,1,1,2)$

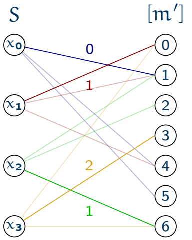


$(0+2+1) \bmod 3 = 0$

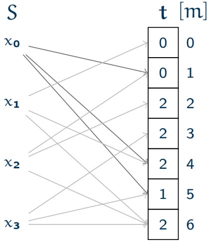
Summary

Performance: e.g. using factor $1.1 > \dfrac{1}{c_0(3, 16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$
  - injective mapping: $m' = m = 1.1 \cdot n \Rightarrow d' = 3$ hash functions for $\mathcal{B}'$
    - $1.1 \cdot n \cdot \lceil \log 3 \rceil$
- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16



v=(0,1,1,2)

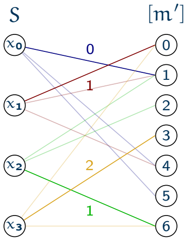


(0+2+1) mod 3=0

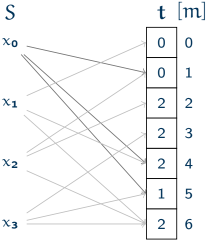
Summary

Performance: e.g. using factor $1.1 > \frac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$
  - injective mapping: $m' = m = 1.1 \cdot n \Rightarrow d' = 3$ hash functions for $\mathcal{B}'$
    - $1.1 \cdot n \cdot \lceil \log 3 \rceil$
    - $1.1 \cdot n \cdot 8/5$ (simple compression)
- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16



$v=(0,1,1,2)$



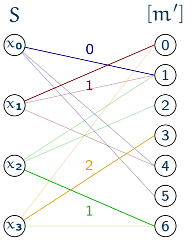
$(0+2+1) \bmod 3 = 0$

Summary

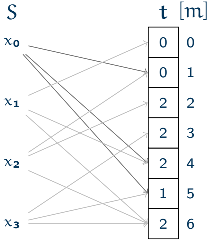
Performance: e.g. using factor $1.1 > \frac{1}{c_0(3,16)}$

- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$
  - injective mapping: $m' = m = 1.1 \cdot n \Rightarrow d' = 3$ hash functions for $\mathcal{B}'$
    - $1.1 \cdot n \cdot \lceil \log 3 \rceil$
    - $1.1 \cdot n \cdot 8/5$ (simple compression)
- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16

Open: Show that c_0 for mixed edge sizes can be arbitrary close to 1.



$v = (0,1,1,2)$



$(0+2+1) \bmod 3 = 0$

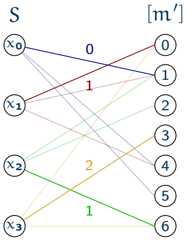
Summary

Performance: e.g. using factor $1.1 > \frac{1}{c_0(3, 16)}$

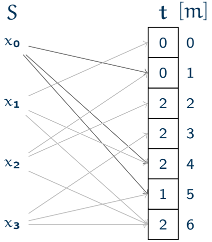
- space utilization
  - retrieval DS: $1.1 \cdot n \cdot \log|V|$
  - injective mapping: $m' = m = 1.1 \cdot n \Rightarrow d' = 3$ hash functions for $\mathcal{B}'$
    - $1.1 \cdot n \cdot \lceil \log 3 \rceil$
    - $1.1 \cdot n \cdot 8/5$ (simple compression)
- construction time: $O(n)$ (expected)
- cell probes: average < 6, worst-case 16

Open: Show that c_0 for mixed edge sizes can be arbitrary close to 1.

Open: Given $\overline{d}$, determine mix of edge sizes that maximizes c_0.



$v = (0,1,1,2)$



$(0+2+1) \bmod 3 = 0$

Thank you!

References I

Bast, H., Mehlhorn, K., Schäfer, G., and Tamaki, H. (2004).
Matching Algorithms Are Fast in Sparse Random Graphs.
In *Proc. 21st STACS*, volume 2996 of *LNCS*, pages 81–92. Springer.

Bohman, T. and Kim, J. H. (2006).
A Phase Transition for Avoiding a Giant Component.
Random Struct. Algorithms, 28(2):195–214.

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.

Cooper, C. (2004).
The Cores of Random Hypergraphs with a Given Degree Sequence.
Random Struct. Algorithms, 25(4):353–375.

References II

Czech, Z. J., Havas, G., and Majewski, B. S. (1992).
An Optimal Algorithm for Generating Minimal Perfect Hash Functions.
Inf. Process. Lett., 43(5):257–264.

Czumaj, A. and Stemann, V. (1997).
Randomized Allocation Processes (Extended Abstract).
In *Proc. 38th FOCS*, pages 194–203. IEEE.

Dietzfelbinger, M., Goerdt, A., Mitzenmacher, M., Montanari, A., Pagh, R., and Rink, M. (2009).
Tight Thresholds for Cuckoo Hashing via XORSAT.
CoRR, abs/0912.0287.

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.

Dietzfelbinger, M. and Pagh, R. (2008).
Succinct Data Structures for Retrieval and Approximate Membership (Extended Abstract).
In *Proc. 35th ICALP (1)*, volume 5125 of *LNCS*, pages 385–396. Springer.

References III

Dietzfelbinger, M. and Rink, M. (2009).
Applications of a Splitting Trick.
In *Proc. 36th ICALP (1)*, volume 5555 of *LNCS*, pages 354–365. Springer, 2009.

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.

Drmota, M. and Kutzelnigg, R. (2012).
A Precise Analysis of Cuckoo Hashing.
ACM Transactions on Algorithms, 8(2):11:1–11:36.

Dubois, O. and Mandler, J. (2002).
The 3-XORSAT Threshold.
In *Proc. 43rd FOCS*, pages 769–778. IEEE Computer Society.

Erdős, P. and Rényi, A. (1960).
On the evolution of random graphs.
Publ. Math. Inst. Hung. Acad. Sci, 5:17–61.

References IV

Fotakis, D., Pagh, R., Sanders, P., and Spirakis, P. G. (2003).
Space Efficient Hash Tables with Worst Case Constant Access Time.
In *Proc. 20th STACS*, volume 2607 of *LNCS*, pages 271–282. 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.

Hopcroft, J. E. and Karp, R. M. (1973).
An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs.
SIAM J. Comput., 2(4):225–231.

Kim, J. H. (2006).
Poisson cloning model for random graphs.
In *Proc. ICM Madrid 2006 Vol. III*, pages 873–898. EMS Ph.

References V

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.

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

Pagh, R. (2001).
On the cell probe complexity of membership and perfect hashing.
In *Proc. 33rd STOC*, pages 425–432. ACM.

Pagh, R. and Rodler, F. F. (2001).
Cuckoo Hashing.
In *Proc. 9th ESA*, volume 2161 of *LNCS*, pages 121–133. Springer.

Rink, M. (2013).
Mixed Hypergraphs for Linear-time Construction of Denser Hashing-based Data Structures.
In *Proceedings 39th SOFSEM*. Springer.
To appear.

References VI

Sanders, P. (2004).
Algorithms for Scalable Storage Servers.
In *Proc. 30th SOFSEM*, volume 2932 of *LNCS*, pages 82–101. Springer.

Wiedemann, D. H. (1986).
Solving Sparse Linear Equations Over Finite Fields.
IEEE Transactions on Information Theory, 32(1):54–62.