On Very Space Efficient Hash Tables

ESA '09: Djamal Belazzougui Fabiano C. Botelho Martin Dietzfelbinger[1]

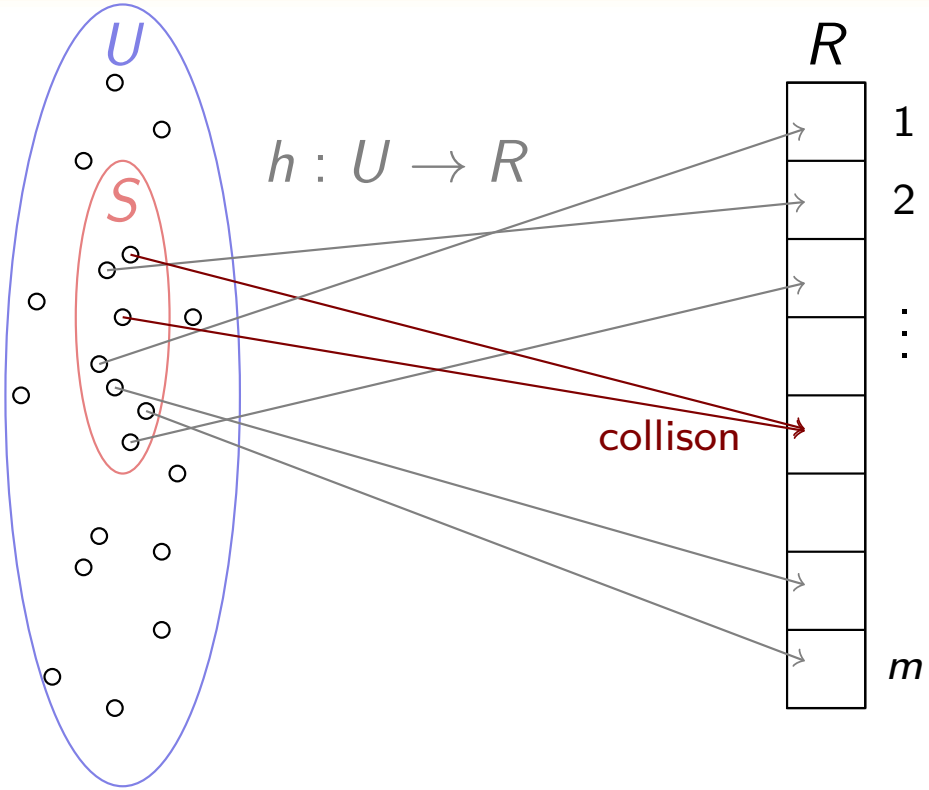
ICALP '10: Martin Dietzfelbinger[1] Andreas Goerdt Michael Mitzenmacher
Andrea Montanari Rasmus Pagh Michael Rink[1]

TU Ilmenau

18.06.2010

[1]Research supported by DFG grant DI 412/10-1.

Hash Table



Map set S of n keys from universe $U, |U| = u$, to range $R, |R| = m$.

Issue: Collisions.

ESA '09:

Hash, Displace, and Compress

(Minimal) Perfect Hash Functions (1)

Perfect hash function (PHF) for $S \subseteq U$:

$$h\colon U \to [m]$$

that is injective on S.

Minimal PHF (MPHF):

$$h\colon U \to [m]$$

that is injective on S, with $|S| = n = m$.
(Bijection, can't do better!)

(Minimal) Perfect Hash Functions (2)

Task

Given $S \subseteq U$, build data structure $D_h = D_h(S)$ such that:
Given $x \in U$, using D_h can calculate $h(x) \in [m]$ in time $O(1)$,
where h is (M)PHF.

Issues:

- Construction time
- Storage space for D_h
  (don't worry about scratch space for construction)
- Evaluation time for h

Clearly a fundamental problem.

Space Lower Bounds

Let $|S| = n$, $|U| = u$, $h : U \to [m]$.

Sharp information theoretic lower bounds [**bits**]:
[Fredman and Komlós, 1984],[Mehlhorn, 1984],[Radhakrishnan, 1992],...

$$\mathcal{S}(h) \geq -(u-n)\log\left(1-\frac{n}{u}\right) - \log(n) + (m-n)\log\left(1-\frac{n}{m}\right)$$

Example ($u \gg n$)

- **PHF** with $m = 1.23n$: $\mathcal{S}(h) \geq 0.89 \cdot n$
- **MPHF** ($m = n$): $\mathcal{S}(h) \geq 1.44 \cdot n$

Long History

- [Fredman et al., 1984] PHF: $\mathcal{S} = O(n \log n), \mathcal{T} = O(n)$, practicable
- [Schmidt and Siegel, 1990] MPHF: $\mathcal{S} = O(n)$
- [Pagh, 1999] MPHF: $\mathcal{S} = (2 + \varepsilon)n \log n$, $\mathcal{T} = O(n)$, practicable
  "Splitting tricks" $\mathcal{S} = \delta n \log n$, for $\delta > 0$ fixed arbitrarily.
- [Hagerup and Tholey, 2001] MPHF: $\mathcal{S} = (\log e + o(1)) \cdot n$,
  $\mathcal{T} = O(n)$, need LARGE n
- [Chazelle et al., 2004] PHF: $\mathcal{S} = O(n)$,$\mathcal{T} = O(n)$, practicable
- [Woelfel, 2006]* MPHF: $\mathcal{S} = O(n \log \log n)$, $\mathcal{T} = O(n)$, dynamic
- [Botelho et al., 2007]* PHF with $m = 1.23n$: $\mathcal{S} < 2n$, $\mathcal{T} = O(n)$
  MPHF: $\mathcal{S} = 2.61n$, $\mathcal{T} = O(n)$, practicable

$\star$: Full randomness assumption.

Results

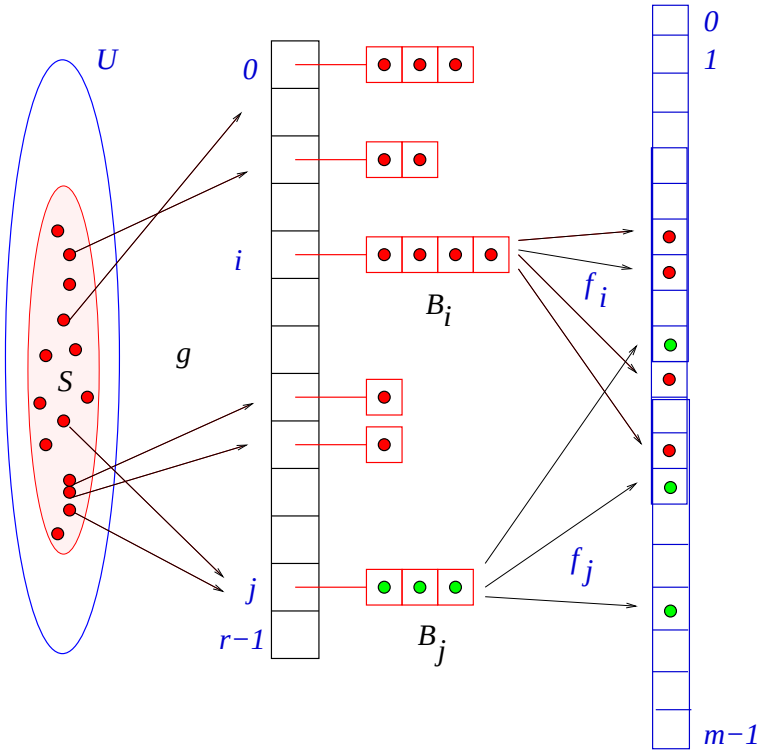
- New algorithm* for PHF: $\mathcal{S} = O(n)$, $\mathcal{T} = O(n)$, practicable.
- Analysis: Potential to get space $(\log e + \delta)n$ for $\delta > 0$ arbitrary.
- Experiments: Best known space usage so far.
  (e.g. $m = 1.01n$ with $1.98n$ bits per key)

$\star$: Full randomness assumption.

Outline

Next . . .

Data Structure



- **First level:** g splits S into buckets $B_i, 0 \leq i < r$.
- **Second level:** For each B_i there is a $f_i : U \to [m]$, "cleverly choosen" from a sequence $(\phi_0, \phi_1, \ldots)$.
- **Overall mapping:**

$$h(x) = f_{g(x)}(x)$$

- **Question:** How to choose the f_i s. t. h is injective on S?
- **Solution:** Modification of "Hash and Displace" by [Pagh, 1999].

Space and Time Usage

Theorem

Let $\lambda = n/r$. Assume $\varepsilon > 0$ is fixed, and $m = (1+\varepsilon)n$. Then

$$\mathrm{E}(\mathcal{S}(h)) = n \log e + n \cdot c_{\mathrm{compress}}(\lambda) + O(\lambda^2) \text{ and}$$
$$\mathrm{E}(\mathcal{T}(h)) = O(n \cdot (2^{\lambda} + (1/\varepsilon)^{\lambda})) \ .$$

Using some suitable compression scheme, e.g. [Fredriksson and Nikitin, 2007]), we get $c_{\mathrm{compress}}(\lambda) < 1$.

Trade-off: Increasing $\lambda = n/r$
(expected size of bucket in intermediate table)

- increases construction time (not evaluation time)
- improves compression and hence decreases space

Next . . .

1. Algorithm
2. **Experiments**

Setup

Hash functions: Instead of fully random functions $\phi_0, \phi_1, \ldots$ use $\phi'_0, \phi'_1, \ldots$ from weaker, efficient class.

Key sets: $2 \cdot 10^7$ unique URLs from the Brazilian Web

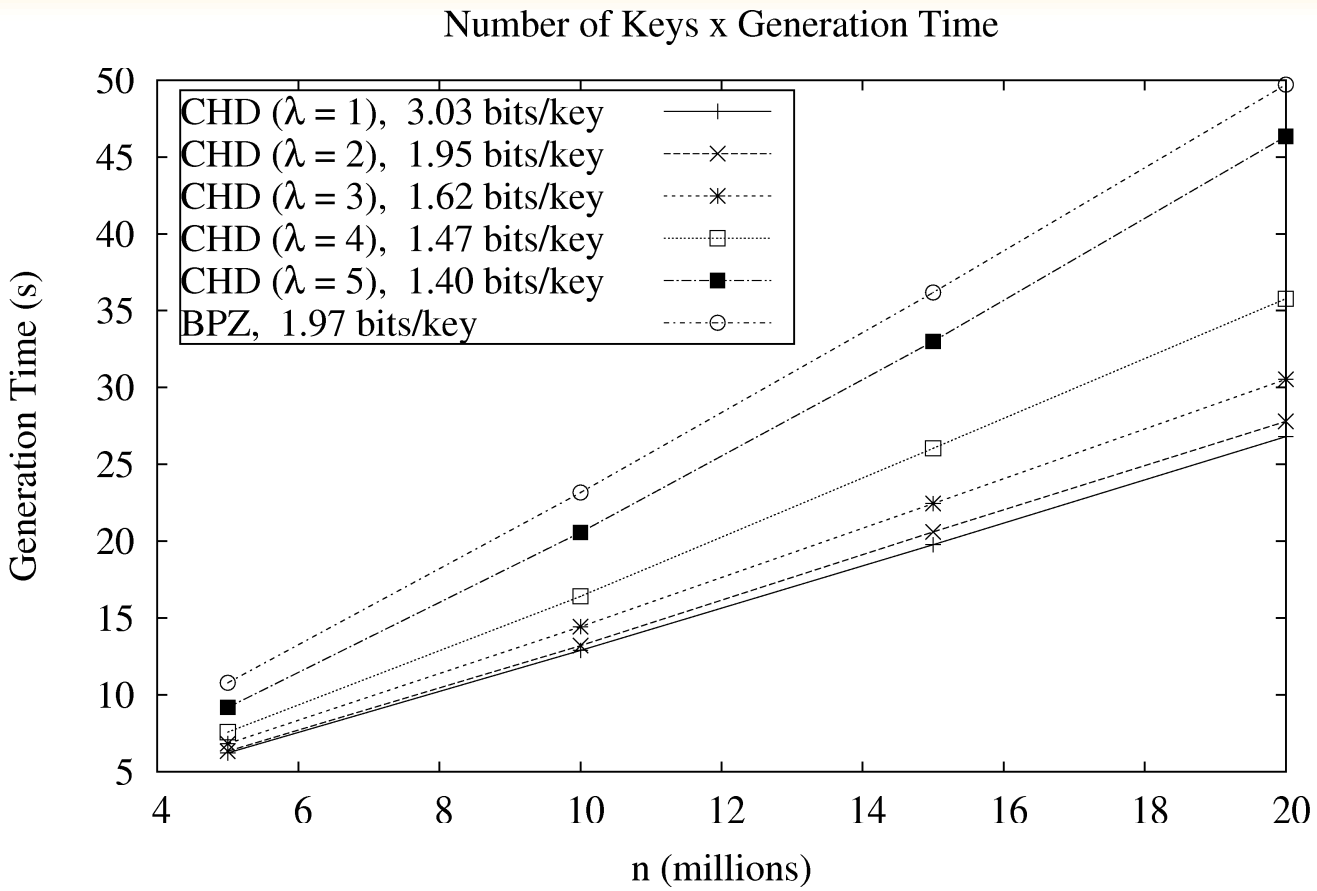
Machine:

- 1.86 Gigahertz Intel Core 2, 4 MByte L2 cache, 4 GByte RAM
- Linux Version 2.6

Source: http://cmph.sf.net (under LGPL).

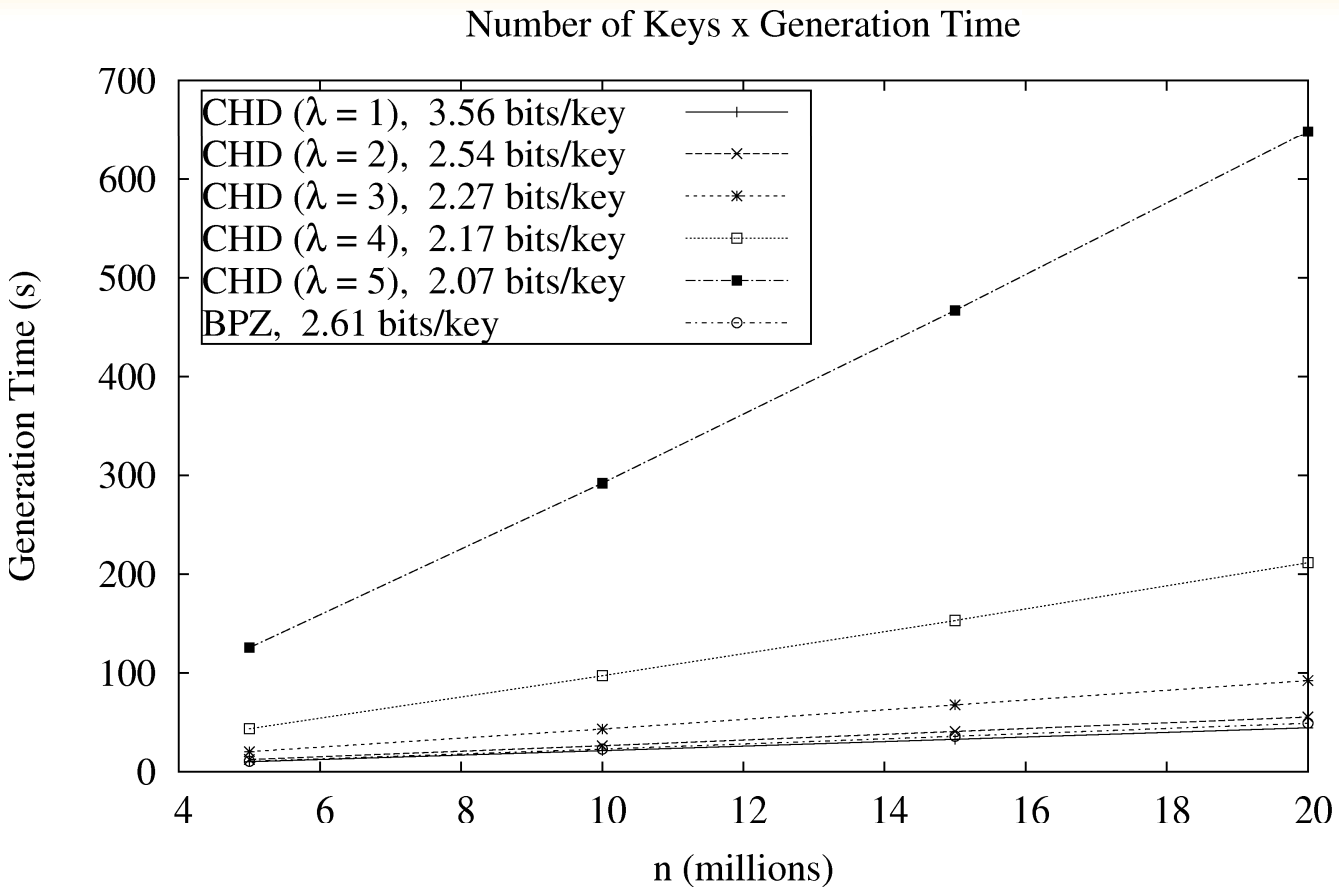
Reference Algorithm: by [Botelho,Pagh,Ziviani, 2007] (BPZ)-main practical perfect hashing algorithm.

PHF



- $m = 1.23n$, space lower bound 0.88 bits/key
- Evaluation time a little higher than for BPZ (ca. 3s for $2 \cdot 10^7$ keys).

MPHF

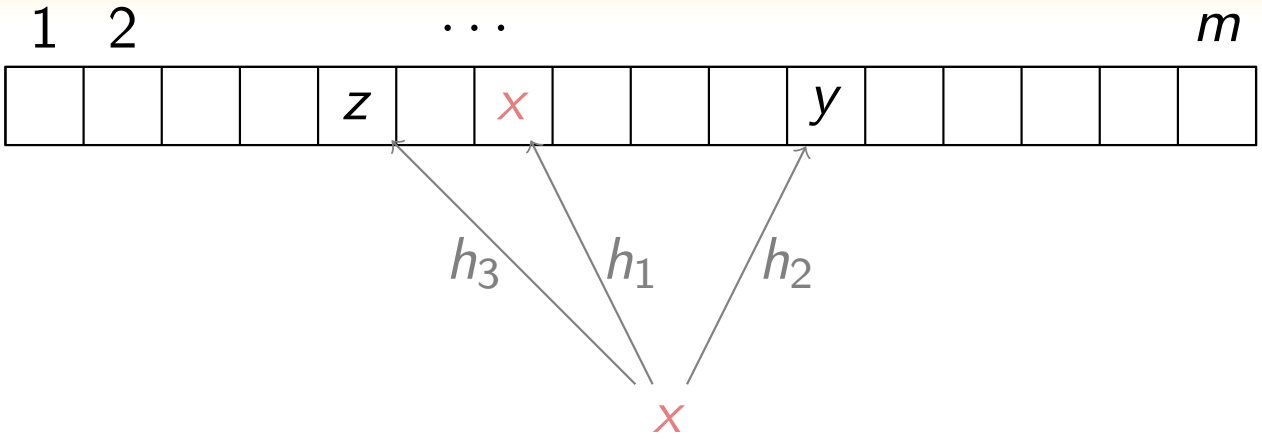


- $m = n$, space lower bound 1.44 bits/key
- Evaluation time a little higher than for BPZ (ca. 4s for $2 \cdot 10^7$ keys).

ICALP '10:

Tight Thresholds for Cuckoo Hashing via XORSAT

k-ary Cuckoo Hashing



[Fotakis, Pagh, Sanders, and Spirakis, 2005]:

- n keys, set $S \subseteq U$
- k hash functions $h_i : U \to [m] = \{1, \ldots, m\}$
- Store x in one of $T[h_i(x)]$, $i = 1, \ldots, k$.
- Maximum one key per cell.

If this is possible for all $x \in S$: Constant lookup time.

Placement

Definition

C.H. works for $(h_i)_{1 \leq i \leq k}$

if there is a injective mapping $\sigma : S \to [m]$

such that $\sigma(x) \in \{h_1(x), h_2(x), \ldots, h_k(x)\}$, for all $x \in S$.

(Can store keys from S with no collision.)

Thresholds (1)

C.H. for $k = 2$: [Pagh and Rodler, 2001, 2004] well understood, works for $\frac{n}{m} < 0.5$.
(Related to appearance of giant connect component in cuckoo graph.)

C.H. for $k \geq 3$:

Theorem ([Fotakis et al., 2005])

There are $C_1 > C_2 > 0$ such that:

$$\frac{n}{m} \leq 1 - e^{-C_2 \cdot k}, m \to \infty \Rightarrow \Pr(\textit{C.H. works}) = 1 - o(1) \ ,$$
$$\frac{n}{m} \geq 1 - e^{-C_1 \cdot k}, m \to \infty \Rightarrow \Pr(\textit{C.H. works}) = o(1) \ .$$

Thresholds (2)

Would like: Sharp Thresholds c_k for $k \geq 3$,
that is $c_k < 1$ such that for all c:

$$\frac{n}{m} \leq c < c_k,\ m \to \infty \quad \Rightarrow \quad \Pr(\text{C.H. works}) = 1 - o(1)\ ,$$
$$\frac{n}{m} \geq c > c_k,\ m \to \infty \quad \Rightarrow \quad \Pr(\text{C.H. works}) = o(1)\ .$$

Outline

Next . . .

1. **Thresholds via XORSAT**
2. Algorithm
3. Experiments

Random k-XORSAT

n clauses, m variables, k literals per clause

Example

$$(\overline{X}_1 \oplus X_2 \oplus \overline{X}_4) \wedge (\overline{X}_2 \oplus \overline{X}_4 \oplus X_5) \wedge (X_3 \oplus \overline{X}_4 \oplus X_5)$$

Question: "Is there an assignment $x = (x_1, \ldots, x_m)$ that gives all clauses value 1?"

Equivalent: Solvability of random sparse system.

$$\mathcal{M}^k_{m,n} \cdot x = b$$

Example (Note: $\overline{X} = 1 \oplus X$)

$X_1 \oplus X_2 \oplus X_4 = 1$ and $X_2 \oplus X_4 \oplus X_5 = 1$ and $X_3 \oplus X_4 \oplus X_5 = 0$

$\mathcal{M}^k_{m,n}$ is an $n \times m$ matrix, 0-1-valued, with exactly k 1's per row, rows chosen independently at random, and $b \in \{0,1\}^n$ is random.

Connections

Known in k-XORSAT research ([Dubois and Mandler, 2002],...):

- $\mathcal{M}^k_{m,n}$ is equivalent to random hypergraph $\mathcal{H}^k_{m,n}$.
- Removing columns with exactly one 1 and the corr. rows does not change the solvability of the system.
- It remains $\hat{n} \times \hat{m}$ matrix $\mathcal{M}^k_{\hat{m},\hat{n}}$ that corr. to the 2-core of $\mathcal{H}^k_{m,n}$.
- Threshold for solvability of random k-XORSAT is where 2-core starts having more edges than nodes.

Results

Main Theorem

Density thresholds for k-ary cuckoo hashing and
random k-XORSAT are the same
and they are at the place where the edge density of the 2-core
of the relevant hypergraph grows beyond 1.

Example

k	3	4	5	6
threshold	0.9179	0.9768	0.9924	0.9974

Simultaneous, independent work:

- [Fountoulakis and Panagiotou, 2009] (arXiv,ICALP '10)
- [Frieze and Melsted, 2009] (arXiv)

Next . . .

A Linear Time Algorithm

Adaption of "selfless-algorithm" by [Sanders, 2004].

Algorithm 1: Generalized Selfless

Input: Hypergraph $\mathcal{H}^k_{m,n} = (V, E)$ with m nodes and n edges.

repeat

  if some node v **has degree** 1 **then**

    choose node v;

  else

    choose node v of minimum priority $\pi(v) = \sum_{e \ni v} \frac{1}{|e|}$;

    if minimum priority > 1 **then**

      return failure;

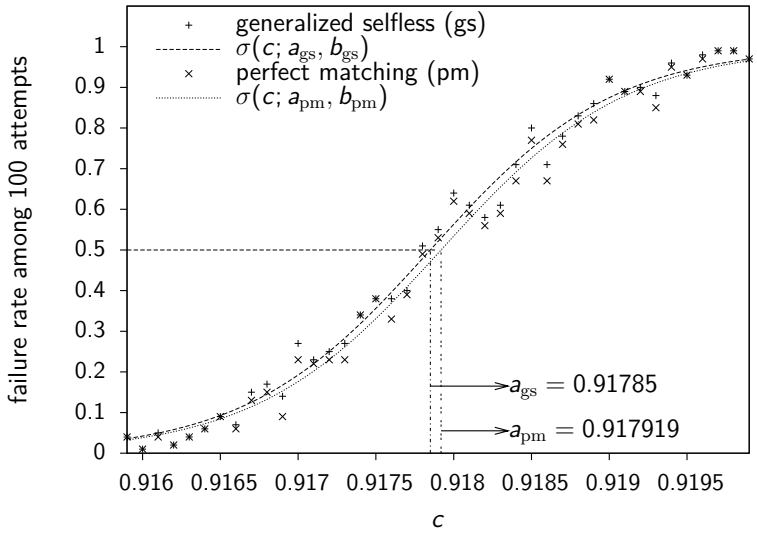
  direct an edge $e \ni v$ with minimal cardinality $|e|$ towards v;

  delete v and e;

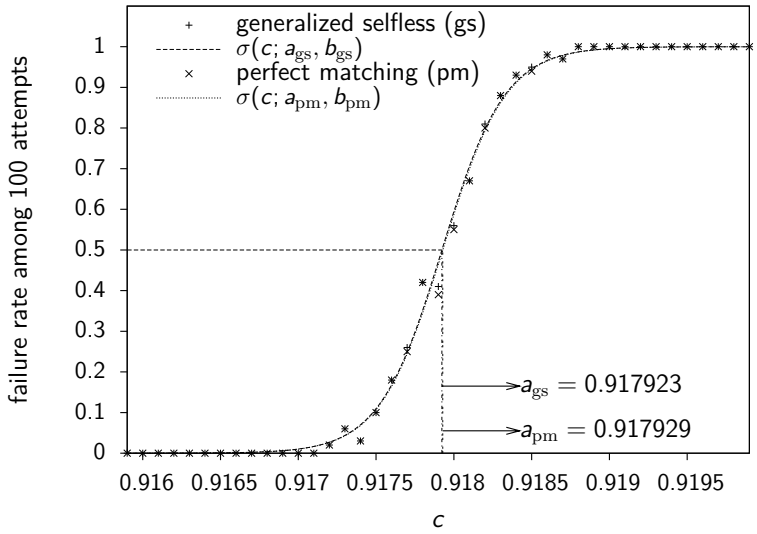
until all edges have been deleted at the end;

Next . . .

Generalized Selfless vs Perfect Matching



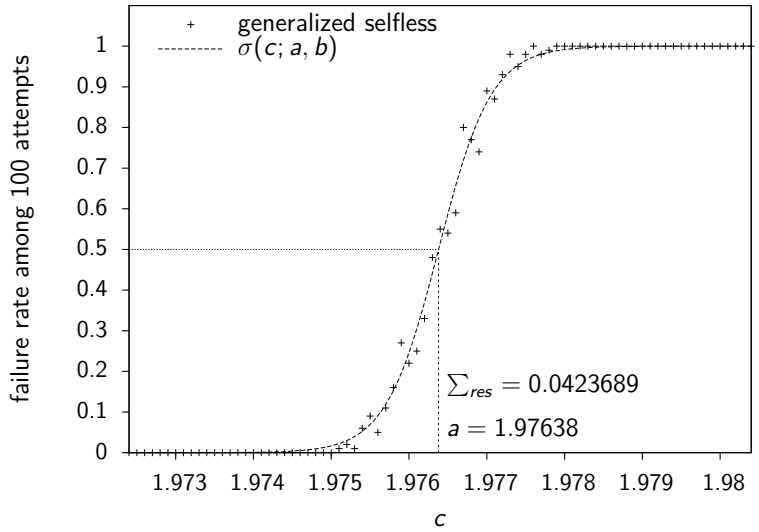
(a) $m = 10^5$



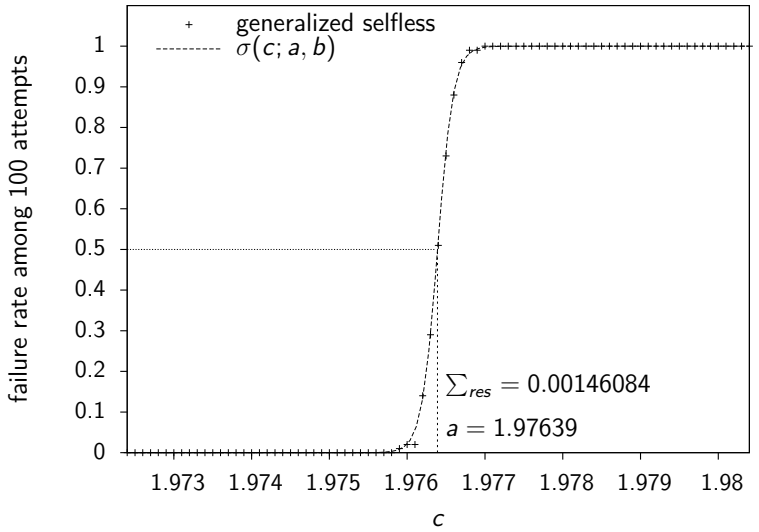
(b) $m = 10^6$

Figure : $k = 3$; theoretical threshold $c_k \approx 0.91794$, interval size 0.004

Larger Buckets



(a) $m = 10^5$



(b) $m = 10^6$

Figure : $k = 3$, $\ell = 2$; **conjectured** threshold value $c_{k,2} \approx 1.97640$

Conjecture proved for all $k \geq 3$ and ℓ sufficiently large by [Gao and Wormald, 2010] (STOC '10).

Non-integer Choices

Theorem

Pr(*C.H works*) *is maximized if for all* $x \in S$ *the expected number of hash values* κ_x *is concentrated on* $\{\lfloor \kappa_x \rfloor, \lfloor \kappa_x \rfloor + 1\}$.

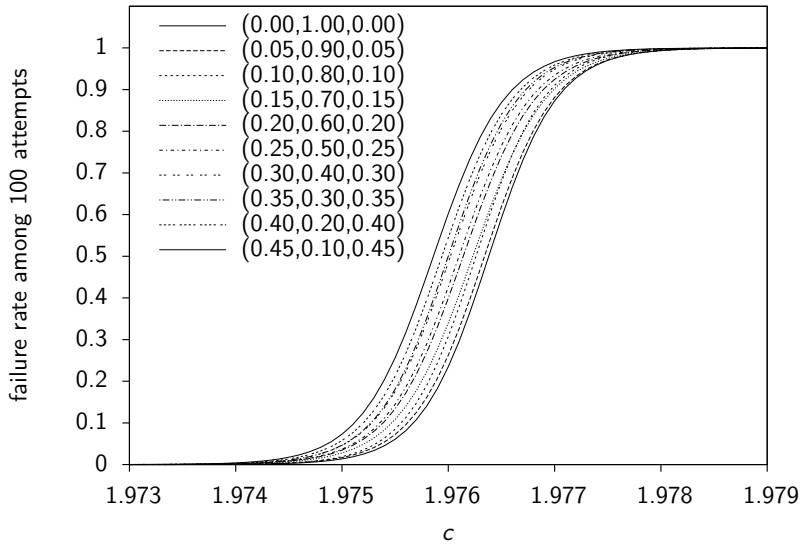


Figure : Various distributions with mean $\kappa_x = 3$; (x, y, z) stands for fraction of keys with $(k = 2, k = 3, k = 4)$; $\ell = 2, m = 10^5$.

Thank you!

Random fit decreasing with random functions

Based on old idea [Tarjan and Yao, 1979].

---

sort buckets B_i according to falling size (permutation π);
initialize vector $V[0..m-1]$ with 0's;
for $i = 0$ **to** $r - 1$ **do**
  for $j = 0, 1, 2, \ldots$ **do**
    $K \leftarrow \{\phi_j(x) \mid x \in B_{\pi(i)}\}$;
    if $|K| = |B_i|$ **and** $K \cap \{k \mid V[k] = 1\} = \emptyset$ **then**
      store index $\sigma(i) = j$;
      forall the $k \in K$ **do** $V[k] \leftarrow 1$;
      break;

Results

Evaluation times:

Algorithms	λ	AllTheWeb ($n = 5 \times 10^6$) Evaluation Time (sec)			URL ($n = 2 \times 10^7$) Evaluation Time (sec)		
		$\alpha = 0.81$	$\alpha = 0.99$	$\alpha = 1.0$	$\alpha = 0.81$	$\alpha = 0.99$	$\alpha = 1.0$
CHD	1	3.53	3.59	4.14	24.06	24.24	26.99
	2	3.41	3.46	4.01	23.24	23.70	26.26
	3	3.40	3.49	4.04	22.71	23.47	26.05
	4	3.42	3.44	4.01	22.58	23.06	25.65
	5	3.43	3.45	4.02	22.41	22.98	25.59
BPZ	1	2.80	–	3.19	19.76	–	22.12

Bibliography (1)

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

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

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

Bibliography (2)

Fredman, M. L. and Komlós, J. (1984).
On the Size of Separating Systems and Families of Perfect Hash Functions.
In *SIAM. J. on Algebraic and Discrete Methods* 5(1), pages 61–68.

Fredman, M. L., Komlós, J., and Szemerédi, E. (1984).
Storing a Sparse Table with $O(1)$ Worst Case Access Time.
In *J. ACM* 31(3), pages 538–544.

Fredriksson, K. and Nikitin, F. (2007).
Simple Compression Code Supporting Random Access and Fast String Matching.
In Proc. 6th *WEA*, pages 203–216.

Hagerup, T. and Tholey, T. (2001).
Efficient Minimal Perfect Hashing in Nearly Minimal Space.
Proc. 18th *STACS*, pages 317–326.

Bibliography (3)

Mehlhorn, K. (1984).
Data Structures and Algorithms 1: Sorting and Searching.
EATCS Monographs on Theoretical Computer Science.

Pagh, R. (1999).
Hash and Displace: Efficient Evaluation of Minimal Perfect Hash Functions.
Proc. 6th *WADS*, pages 49–54.

Radhakrishnan, J. (1992).
Improved Bounds for Covering Complete Uniform Hypergraphs.
In *Inf. Process. Lett.* 41(4), pages 203–207.

Schmidt, J. P. and Siegel, A. (1990).
The Spatial Complexity of Oblivious k-Probe Hash Functions.
In *SIAM J. Comput.* 19(5), pages 775–786.

Bibliography (4)

Tarjan, R. E. and Yao, A. C.-C. (1979).
Storing a Sparse Table.
Commun. ACM 22(11), pages 606–611.

Vigna, S. (2008).
Broadword Implementation of Rank/Select Queries.
In Proc. 7th *WEA*, pages 154–168.

Woelfel, P. (2006).
Maintaining External Memory Efficient Hash Tables.
In Proc. 9th/10th *APPROX/RANDOM*, pages 508–519.

Thresholds

$\ell \backslash k$	2	3	4	5	6
2	–	0.9179352767	0.9767701649	0.9924383913	0.9973795528
3	1.7940237365	1.9764028279	1.9964829679	1.9994487201	1.9999137473
4	2.8774628058	2.9918572178	2.9993854302	2.9999554360	2.9999969384
5	3.9214790971	3.9970126256	3.9998882644	3.9999962949	3.9999998884
6	4.9477568093	4.9988732941	4.9999793407	4.9999996871	4.9999999959
7	5.9644362395	5.9995688805	5.9999961417	5.9999999733	5.9999999998

A Linear Time Algorithm (1)

Adaption of "selfless-algorithm" [Sanders, 2004].

Algorithm 2: (k, ℓ)-Generalized Selfless

Input: Hypergraph $\mathcal{H}^k_{m,n} = (V, E)$ with m nodes and n edges.

for $t \leftarrow 1$ **to** n **do**

- $V_0 \leftarrow \{v \in V : v \text{ is incident to undirected edge}\}$;
- $E_0 \leftarrow \{e \in E : e \text{ is undirected}\}$;
- find $v \in V_0$ with smallest priority $\pi(v)$;
- **if** $\pi(v) > \ell$ **then return failure**;
- ;
- choose $e \in E_0 \cap \{e : v \in e\}$ with minimum weight $\omega(e)$;
- direct e towards v

A Linear Time Algorithm (2)

- $\mathcal{D}(v)$ set of hyperedges directed towards node v
- $\mathcal{U}(v)$ set of undirected hyperedges incident to node v

Edge weight:

$$\omega(e) \leftarrow |\{v \in e : |\mathcal{D}(v)| < \ell\}|$$

Node priority:

$$\pi(v) = \begin{cases} 0, & \text{if } |\mathcal{U}(v)| + |\mathcal{D}(v)| \leq \ell \\ \sum_{e \in \mathcal{U}(v)} \frac{1}{\omega(e)} + |\mathcal{D}(v)|, & \text{otherwise} \end{cases}$$

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

Cain, J. A., Sanders, P., and Wormald, N. C. (2007).
The Random Graph Threshold for k-orientiability and a Fast Algorithm for Optimal Multiple-Choice Allocation.
In Proc. 18th *SODA*, pages 469–476.

Calkin, N. J. (1997).
Dependent Sets of Constant Weight Binary Vectors.
In *Combinatorics, Probability & Computing* 6(3), pages 263–271.

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

Bibliography (2)

Creignou, N. and Daudé, H. (2003).
Smooth and sharp thresholds for random k-xor-cnf satisfiability.
In *ITA* 37(2), pages 127–147.

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

Fernholz, D. and Ramachandran, V. (2007).
The k-orientability Thresholds for $G_{n,p}$.
In Proc. 18th *SODA*, pages 459–468.

Fotakis, D., Pagh, R., Sanders, P., and Spirakis, P. G. (2005).
Space Efficient Hash Tables with Worst Case Constant Access Time.
In *Theory Comput. Syst.* 38(2), pages 229–248.

Bibliography (3)

Fountoulakis, N. and Panagiotou, K. (2009).
Sharp Load Thresholds for Cuckoo Hashing.
In *CoRR*, abs/0910.5147.

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

Gao, P. and Wormald, N. C. (2010).
Load Balancing and Orientability Thresholds for Random Hypergraphs.
In Proc. 42nd *STOC*, pages 97–104.

Molloy, M. (2005).
Cores in Random Hypergraphs and Boolean Formulas.
In *Random Struct. Algorithms* 27(1), pages 124–135.

Bibliography (4)

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

Pagh, R. and Rodler, F. F. (2004).
Cuckoo hashing.
In *J. Algorithms* 51(2), pages 122–144.

Sanders, P.
Algorithms for Scalable Storage Servers.
In Proc. 30th *SOFSEM*, pages 82–101.